OPTIMAL CONSTELLATION DESIGN FOR ORBITAL MUNITIONS

DELIVERY SYSTEM

THESIS

Jason Anderson, Major, USAF

AFIT/GSS/ENY/04-M01

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GSS/ENY/04-M01

OPTIMAL CONSTELLATION DESIGN FOR ORBITAL MUNITIONS DELIVERY SYSTEM

THESIS

Presented to the Faculty

Department of Aeronautics and Astronautics

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science (Space Systems)

Jason Anderson, BS

Major, USAF

March 2004

AFIT/GSS/ENY/04-M01

OPTIMAL CONSTELLATION DESIGN FOR ORBITAL MUNITIONS DELIVERY SYSTEM

Jason Anderson, BS

Major, USAF

Approved:

___________/SIGNED/________________ ________
Steven G. Tragesser, PhD (Chairman) Date

___________ /SIGNED/________________ ________
William E. Wiesel, PhD (Member) Date

___________/SIGNED/________________ ________
Donald L. Kunz, PhD (Member) Date

AFIT/GSS/ENY/04-M01

Abstract

Rather than delivering conventional munitions through the airspace of uncooperative nations, a constellation of space-stored weapons could potentially target any point on the Earth and arrive within the time it takes to de-orbit and re-enter through the atmosphere. The research involves applying the dynamics of atmospheric re-entry to a Common Aero Vehicle (CAV) and defining a 'footprint' of attainable touchdown points. The footprint is moved forward to create a swath representing all the possible touchdown points in a 90 minute window. A nominal constellation of CAVs is established using a 'streets of coverage' technique, and both analytic studies and numeric genetic algorithm techniques are used to modify the nominal constellation. A minimum number of CAVs is identified which ensures payload delivery to an area of interest within 90 minutes.

Acknowledgments

First, I would like to extend my sincere appreciation to my faculty advisor, Dr. Steve Tragesser, for his guidance and support throughout the course of this thesis effort. Without his patience, insight, scientific expertise, and wisdom, I would have not succeeded in this endeavor. I must also acknowledge several of my classmates, whose abundant humor, constant encouragement, and endless patience in explaining things kept me motivated and engaged: Mr. Matthew Press, Capt. Dennis McNabb, Capt. Stan Straight, Capt. Mark Faulstich, Capt. Steve Lindemuth, Capt Todd Hoover, Flt Lt Matt Colbert, and Maj. Martin Lindsey. I would also like to thank my sponsor, Lt Col Ed Phillips, from the National Security Space Architect, for both the support and latitude provided to me on this project. Thank you also to my children, whose laughter and love are always in abundant supply. Finally, and most importantly, my deepest love and admiration for my wife, without whom I would have nothing of value, and who is the reason I am here in the first place.

Jason Anderson

Table of Contents

Page

Page

List of Figures

Page

List of Tables

Page

OPTIMAL CONSTELLATION DESIGN FOR ORBITAL MUNITIONS DELIVERY SYSTEM

I. Introduction

Background

The Common Aero Vehicle (CAV) is a lifting body capable of atmospheric re-entry (1:29). This weapon platform could be deployed on air-launched suborbital missiles, ICBMs, or launched into low Earth orbit via conventional boosters. The CAV is envisioned to be self-guiding toward its target, using inertial and possibly GPS navigation in concert with aerodynamic controls. When placed in orbit around the Earth, it could be used to deliver a munitions payload to any location within its re-entry footprint. Furthermore, a constellation of such vehicles could give 100% delivery coverage over any desired portion of the Earth's surface.

Problem Statement

In response to a query by the National Security Space Architect (NSSA), we will attempt to quantify, both analytically and numerically, the minimum number of CAVs required to fully cover a given portion of the Earth. Terrestrial delivery is required to occur no later than 90 minutes from the time a decision is made to strike a target. Coverage may be any band of latitude, extending from 0° to the latitude of interest.

Research Objectives/Focus

The research involves several disciplines and will determine optimal solutions for constellations of CAVs, dependent upon several design parameters. An exploration of atmospheric re-entry is necessary to determine the touchdown footprint of a single CAV. Analytic constellation design will be used extensively to define several types of baseline constellations. Numeric genetic algorithm (GA) techniques will be used to search for non-analytic solutions. Finally, we will compare the results of both techniques and identify the most efficient types of constellations to use in this application.

Methodology

While some of the research involves analytical evaluation of CAV constellations, a great deal of the work depends upon the results of numeric simulation. Footprint width, a fundamental quantity used in the analysis, is solely determined from numeric integration of the CAV's equations of motion. Additionally, generation of Earth coverage statistics as well as the entire GA routine is numeric in nature. All of these numeric techniques are carried out using MATLAB© (10), with the GA routine using an add-on software package from Optimal Synthesis© (11).

Assumptions/Implications

Since this work represents a first look at this combining atmospheric re-entry and constellation design, there are several basic assumptions which were made in order to reduce the computational complexity of the problem and obtain a first-order solution. First, the Earth is assumed to be spherical and non-rotating. Second, the atmosphere is

assumed to be exponential. Third, gravity is assumed to be constant throughout the CAV's trajectory. Fourth, we assume the CAV does not have any delta-v capability other than that required to de-orbit. Finally, the CAV is not placed under any heating, dynamic loading, or g-force constraints. Application of these assumptions leads to a more conservative design than might be possible using more complex techniques.

Preview

Analytic results point to a polar inclined streets of coverage (SOC) constellation as being the most efficient way to obtain 100% coverage for high latitudes. However, GA techniques reveal a modified, inclined SOC constellation that, when investigated further, can be obtained analytically and provides an improvement over the polar SOC constellation when certain coverage requirements are imposed.

II. Literature Review

Overview

There has been a great deal of research in the disciplines of constellation design, genetic algorithm (GA) search techniques, and atmospheric re-entry. In some cases, GA techniques have been applied to satellite constellations (2:169-77), but the two disciplines of atmospheric re-entry and constellation design have generally been treated separately.

Relevant Research

Much of the research in constellation design has focused on minimizing the number of communications or remote sensing satellites required to continuously cover at least some portion of the Earth (3, 4:179-84, 5:31-64, 6, 7:1419-30). These works all begin with the direct relationship between swath width and satellite altitude. Satellites are assumed to have circular footprints, and analysis consists of examining the number of orbit planes and the number of satellites per plane as variables leading to the determination of swath width. Once swath width is obtained and the constellation is minimized, the required altitude can be directly calculated. Conversely, constraints on orbit altitude may dictate a swath width, which can then be used to find a minimum number of satellites that yield the desired coverage.

Constellation design using the streets of coverage (SOC) approach has also been investigated. In this method, the ascending nodes of orbit planes are evenly spaced through 360° for arbitrarily inclined constellations, and through 180° for polar inclined constellations (7:1420, 8:188-200, 9:431-33). These works arrange satellites such that

their circular footprints are aligned in such a way as to minimize the number or orbit planes required. This is generally done by placing the 'dip' created by adjacent circular footprints next to the 'bulge' created by a footprint in the next orbit plane. However, because the last orbit plane is counter-rotating with respect to its next neighbor, its nodal spacing must be smaller than the average spacing (7:1420-23). In any case, none of these works investigates nodal spacing for values other than 180° or 360°.

Some research has also focused on determining the intersection of both co-rotating and counter-rotating swaths in order to facilitate coverage of specific latitudes or latitude bands (3:8, 5:62-3). An analytic method of determining the latitude of swath crossings is developed using spherical geometry. We refer to this analysis extensively in the analytic portion of this work.

Genetic algorithm search techniques are widely researched and documented. No new techniques are presented here; rather, standard GA search techniques (12:211-15) are employed with the aid of a MATLAB© (10) add-on software package from Optimal Synthesis© (11). We refer the reader to texts by Holland (17) and Koza (18) for more information on genetic algorithms in general.

Atmospheric re-entry is also a well-researched subject. Many theoretical and practical studies have been conducted on hypersonic re-entry vehicle dynamics and control. Generally, these studies have focused on recovering manned spacecraft (13:239-68) or on ballistic re-entry of ICBM warheads (14:8-16). Of specific interest, however, is a controllable re-entry vehicle's footprint of possible touchdown points. This topic has

been addressed, and the footprint has been analytically determined (15:207-10). The results of this particular work are critical to the problem addressed in this study.

Applicability of Current Research

This research was sponsored by the National Security Space Architect (NSSA) in response to a query regarding potential offensive space architectures. The solutions presented in this paper represent a first look at this problem from the standpoint of storing munitions on-orbit.

This problem differs from previous research in that swath width is no longer a function of altitude or the number of satellites per plane, but rather a fixed value determined solely by the re-entry performance of the CAV. Furthermore, the system does not operate instantaneously as with remote sensing or communications platforms. CAVs cannot deliver their payloads until they have physically passed through the atmosphere, which consumes a finite amount of time.

Therefore, there is a specific requirement on the time until delivery. This allows us to account for both re-entry time and spacing of the CAVs within an orbit plane. In this problem, delivery must be within 90 minutes from the time of de-orbit.

We will also investigate SOC constellations in which nodal spacing takes on some value between 180° and 360°. This approach, combined with the non-continuous coverage, creates a unique problem to solve.

III. Methodology

Overview

We begin by simulating the equations of motion for the CAV during atmospheric re-entry to obtain a maximum lateral distance and the time to attain this distance. With this information we define the area that the munitions could impact within 90 minutes. The remainder of the problem consists of arranging a constellation of CAVs such that their touchdown swaths completely cover the Earth.

Much of the work was numerical in nature, and several functions were created by the author to aid in processing data. They include a re-entry profile function, which simulates the equations of motion and outputs latitude and longitude as a function of time; a constellation development function, which creates a nominal constellation of CAVs based on inputs such as swath width, inclination, and desired latitude coverage; and an Earth grid function which is used to calculate Earth coverage statistics. The GA portion of the analysis relied heavily on a fitness function, which incorporates the number of CAVs in a constellation and the percentage of Earth coverage generated by the constellation to produce a fitness value relative to all other constellations being considered. Finally, the GA algorithm used many built-in functions included in a MATLAB© add-on package from Optimal Synthesis©. See Appendix C for the MATLAB© code used in these functions.

Atmospheric Re-entry

We begin by defining the reference frame in which the CAV operates. Starting from an inertial frame X-Y-Z, with its origin at Earth center, we introduce a rotating frame x-y-z, also with its origin at Earth center. This frame is rotated through two angles: Earth east longitude, θ, and Earth latitude, φ. The CAV's position vector lies along the x-axis. A third frame a-b-c, also rotating, is centered on the CAV. The a-b plane lies in the local vertical plane, with the b-axis directed out the front of the CAV. The c-axis is given by $c = a \times b$. From this frame we define the flight path angle, γ, measured downward from the local horizontal to the velocity vector; the heading angle, Ψ, measured from the local latitude to the projection of the velocity vector onto the local horizontal; and the bank angle, σ, measured from the a-b plane to the lift vector. We also note that the lift vector is always perpendicular to the velocity vector. Figure 1 shows these relationships.

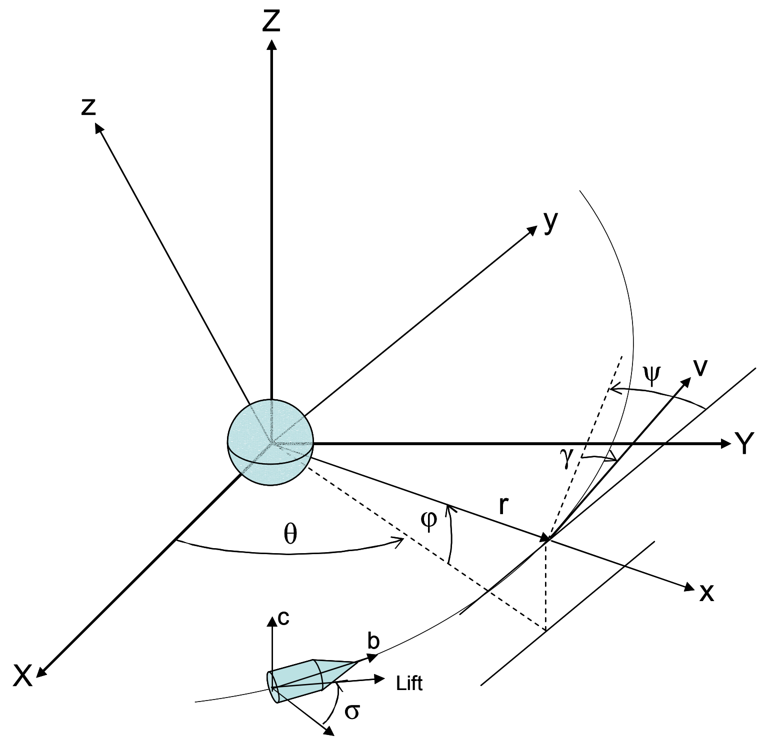


Figure 1 CAV Coordinate Systems

We make some simplifying assumptions before proceeding. The Earth is assumed to be spherical and non-rotating. Additionally, the atmosphere is assumed to be exponential, and gravity is assumed constant throughout the trajectory. Based on these assumptions and reference frames, the equations of motion for atmospheric re-entry are as follows (16):

$$
\begin{aligned}
\dot{h} &= -v\sin\gamma \\
\dot{\theta} &= \frac{v\cos\gamma\cos\Psi}{\cos\phi(R_E + h)} \\
\dot{\phi} &= \frac{v\cos\gamma\sin\Psi}{(R_E + h)} \\
\dot{v} &= -\frac{D}{m} + g\sin\gamma \\
-v\dot{\gamma} &= \frac{v^2}{(R_E + h)}\cos\gamma + \frac{L}{m}\cos\sigma - g\cos\gamma \\
v\cos\gamma\,\dot{\Psi} &= \frac{v^2}{(R_E + h)}\cos^2\gamma\sin\Psi\tan\phi - \frac{L}{m}\sin\sigma
\end{aligned}
\tag{1}
$$

We refer the reader to the section on notation for explanation of these variables.

The lift vector, as the shaping force of the re-entry trajectory, is controlled by the bank angle, σ. This is a similar approach to that used in the Space Shuttle program (13). Starting from a point immediately after the re-entry burn, a footprint of possible impact points is constructed. The maximum downrange capability is obtained by maximizing lift and holding bank angle constant at 0°. Lateral range is obtained by commanding bank angle to some value other than 0° in order to give the lift vector a horizontal component, which then turns the vehicle through its descent. Optimal control of bank angle in maximizing lateral range has been investigated (15:208), but for this effort we choose a constant bank angle to simplify the process. Vinh gives 45° as a suboptimal constant value, but also notes that for any given lift-to-drag ratio, a value greater than 45° will produce the greatest lateral range. This optimal value is obtained by solving the cubic equation

$$\frac{E^4}{8}\left(1-\frac{6}{\pi^2}\right)\alpha^3+E^2\left(1-\frac{E^2}{16}\right)\left(1-\frac{6}{\pi^2}\right)\alpha^2+\left[\frac{1}{4}\left(8-E^2\right)-\frac{3}{4}E^2\left(1-\frac{6}{\pi^2}\right)\right]\alpha-1=0, \qquad (2)$$

where $E = C_d/C_l$ and $\alpha = \cos^2\sigma$ (16:353).

Once bank angle is obtained, the equations of motion are numerically integrated in MATLAB© using a variable step size 5th order Runge-Kutta algorithm. Of particular interest in this application is the CAV's crossrange, or lateral capability. The maximum lateral range, λ_{max}, is primarily a function of bank angle and vehicle ballistic parameters.

Although control in this simulation is open loop, we choose to employ a simple method of control to help maximize lateral range. In this scheme the CAV maintains its optimum bank angle (from Equation 2) until the heading angle is turned 90° away from the initial heading, at which time the bank angle is set to 0°. This prevents the CAV's trajectory from becoming a spiral and allows a greater lateral range than if the bank angle were fixed throughout the trajectory. We also obtain the time to reach λ_{max}, denoted as $t_{re\text{-}entry}$, and the downrange distance of λ_{max}, denoted as $d_{re\text{-}entry}$.

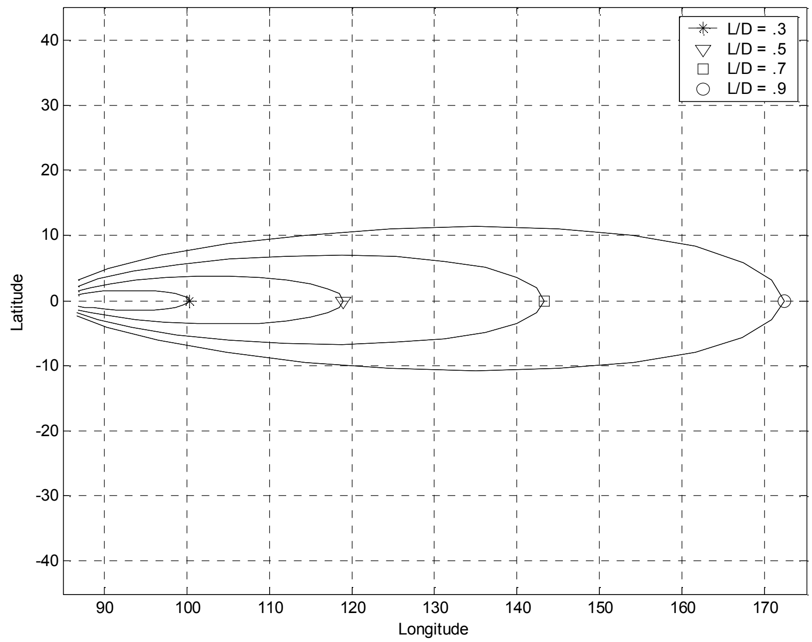


Figure 2 Footprint Size vs. Lift-to-Drag Ratio

There are several quantities from Equation 1 which might change the outcome of the simulation. Chief among these are the ballistic quantities associated with the CAV itself: its mass, frontal surface area, and its coefficients of lift and drag. We assume a fixed mass and frontal area, and focus instead on the effects of the ratio of lift to drag, denoted as *L/D*. Figure 2 illustrates the effect of *L/D* on footprint size and displacement from a de-orbit burn at 0° latitude and 0° longitude.

The left ends of the footprints are open due to the fact that we do not allow bank angle to change with time (e.g. performing roll reversals). The complete footprint can be obtained using more sophisticated methods (15:207-10), but for this application we are only interested in the maximum width of the footprint. Table 1 lists some possible values

for L/D and the associated values for lateral range, λ_{max}, time of re-entry, $t_{re\text{-}entry}$, and downrange distance of λ_{max}, $d_{re\text{-}entry}$.

Table 1 L/D and Swath Dimensions

L/D	λ_{max}	$t_{\text{re-entry}}$ (sec)	$d_{re\text{-}entry}$
0.3	1.52°	1921	94.16°
0.5	3.75°	2186	103.77°
0.7	7.00°	2518	115.94°
0.9	11.25°	2907	129.91°

Analytic Constellation Design

We begin the analysis by defining the total area which can be covered by a single CAV within the 90 minute time constraint. Since our starting point can be anywhere in the CAV's orbit, we define a swath of coverage based on the current position of the CAV within its orbit, u_0; the orbital mean motion, ω; and the quantities λ_{max}, $t_{re\text{-}entry}$, and $d_{re\text{-}entry}$.

The swath length is defined as follows. The CAV can attain any point within the footprint, but we are interested only in the points at which the swath is at its widest. Therefore, we do not consider points before or after $d_{re\text{-}entry}$. This omission gives a conservative estimate of the ground swath (as discussed below), but greatly simplifies the analysis. The closest point along the ground trace is given by $u_0 + d_{re\text{-}entry}$. If we allow the CAV to travel through its orbit until the last possible moment, defined by $90 - t_{re\text{-}entry}$, we obtain the furthest point along the ground trace that the CAV can attain. The swath

consists of the area between these two endpoints. Figure 3 illustrates the relationship between time and distance and the CAV's swath length.

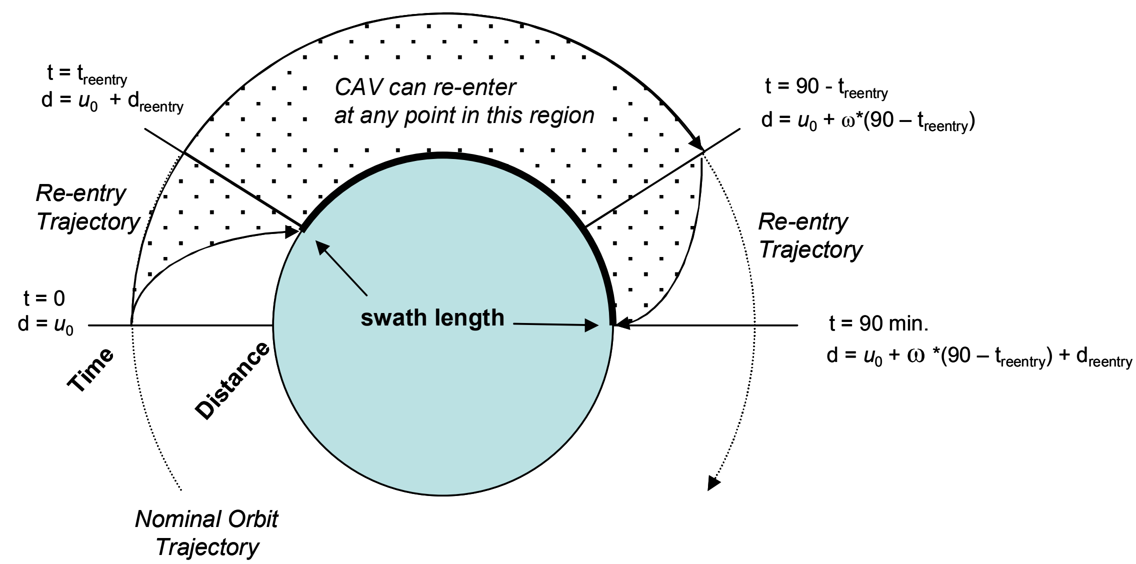


Figure 3 CAV Swath Length

Although the CAV is capable of attaining any point in the footprint, the time it takes to travel to that point is variable. To simplify the analysis, we choose a constant value for $t_{re\text{-}entry}$. Figure 4 illustrates the time difference between two points in an example footprint, with *L/D* at 0.7. The difference between the banked trajectory, which takes 2518 seconds, and the straight trajectory, which takes 2562 seconds, represents only a 1.7% deviation. The time difference grows with increased L/D; the difference is approximately 9% at an L/D of 1.2. We will always use the longer time to represent $t_{re\text{-}entry}$ in order to maintain a conservative design.

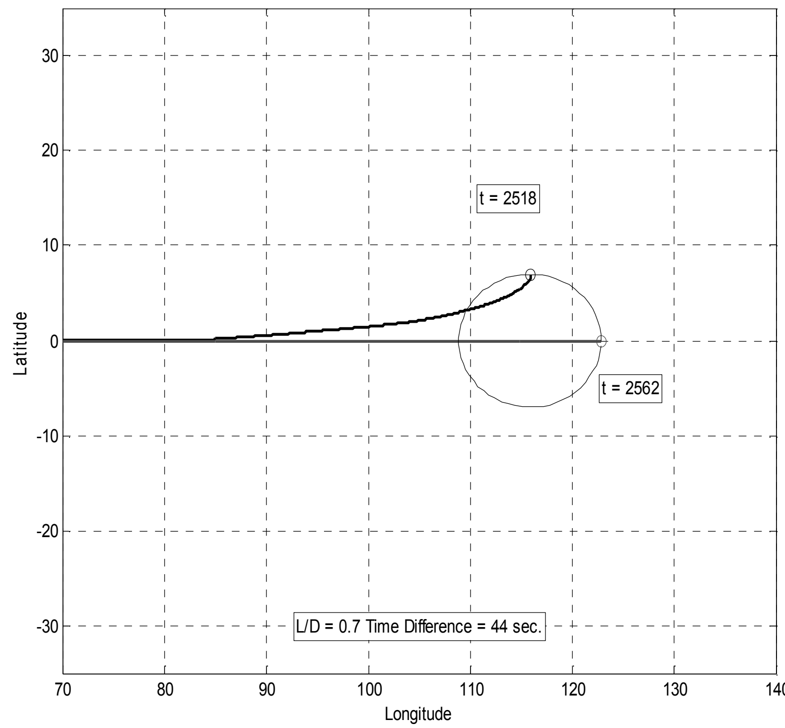


Figure 4 Time Difference for Straight vs. Banked Trajectory

The ends of the swath are somewhat irregular in shape due to the possibility of using varying amounts of bank angle during the descent (see Figure 2). In this application, however, we assume that the swath is of constant width *w*, where $w = 2\lambda_{\max}$. Additionally, we note that the length of the swath is given by

$$l = \omega(90 - t_{re-entry}) \tag{3}$$

Although this approach does not maximize the full potential of the CAV, it allows for a simpler analysis of constellation coverage. Figure 5 and Figure 6 illustrate this concept.

Now, based on the length of a swath of coverage, we can directly calculate the number of CAVs required per orbit plane by

$$s = ceiling\left(\frac{2\pi}{l}\right), \tag{4}$$

where l is given in Equation 3 and *ceiling* is a function that rounds up to the nearest integer. Since s must be an integer, there will likely be some level of overlap between the ends of the individual swaths within the orbit plane.

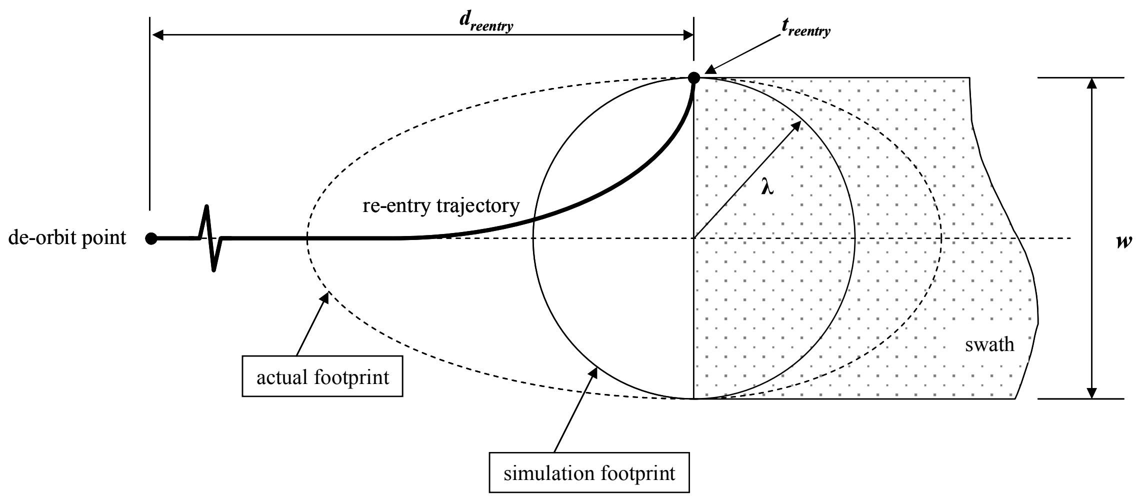


Figure 5 Actual vs. Simulated Footprint Comparison

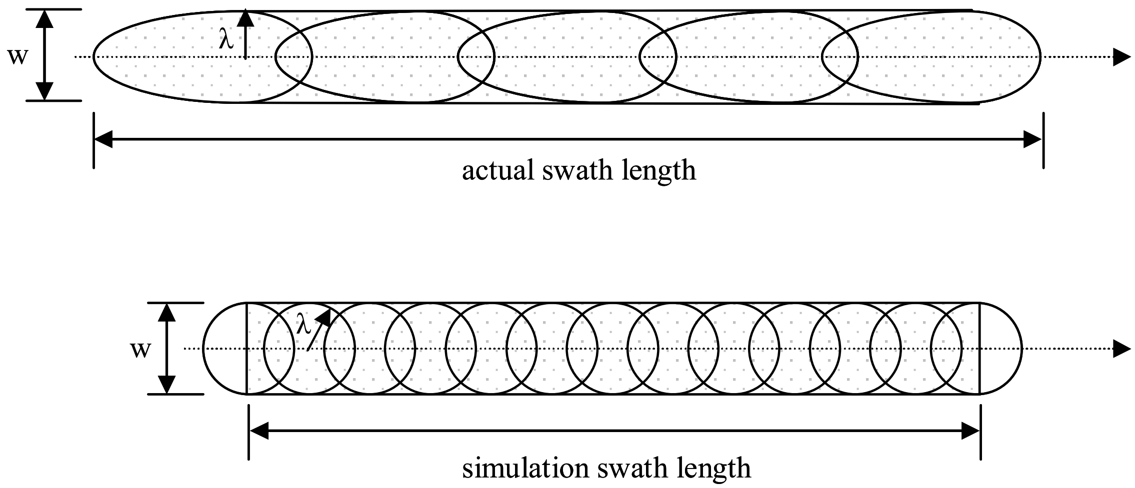


Figure 6 Actual vs. Simulated Swath Length Comparison

Streets of Coverage Constellation Design

Given that we have a continuous swath of coverage of width *w* for each orbit plane, our task is to arrange the planes such that we cover the desired portion of the globe in the most efficient fashion. To begin, we adopt a streets of coverage (SOC) approach, in which we ensure equatorial coverage by setting adjacent orbit planes close enough so that they leave no gaps at the equator (8:191-93, 9:431-33). SOC constellations may be arbitrarily inclined, in which case the ascending nodes are equally spaced through 360°. They may also be polar inclined, in which case the ascending nodes are equally spaced through 180° (9:431). We will investigate both these options as well as a third, modified type of SOC constellation in which the ascending nodes are equally spaced through some value between 180° and 360°.

For inclined SOC constellations, we are free to choose any inclination for the orbit planes as long as the required latitudes remain covered. We also note that as the orbit planes become more inclined, the swath will cover a larger portion of the equator, thereby reducing the total number of planes required to cover the entire equator. Figure 7 illustrates this concept and shows the swath as it crosses the equator.

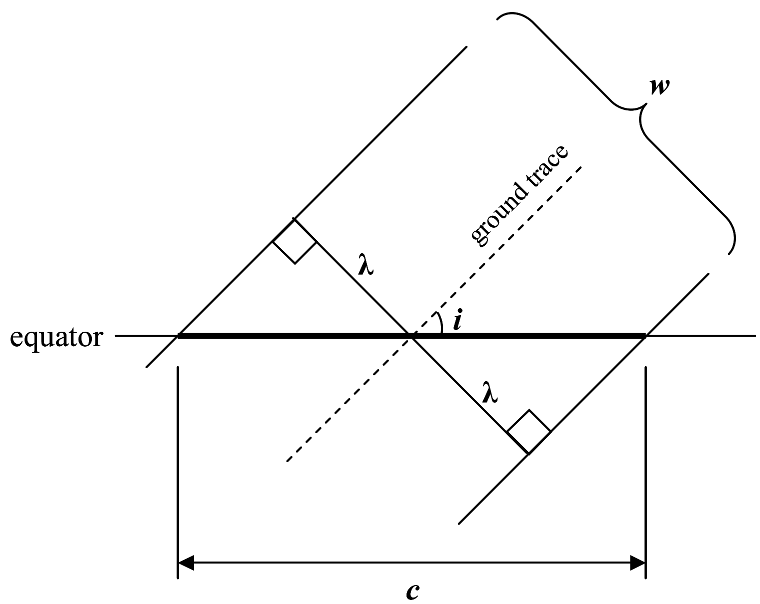


Figure 7 Swath Width at Equator Crossing

Application of spherical trigonometry gives the crossing width by

$$c = \sin^{-1}\left(\frac{\sin w}{\sin i}\right). \tag{5}$$

The number of orbit planes required to produce an inclined SOC constellation is given by

$$p = ceiling\left[\frac{2\pi}{c}\right]. \tag{6}$$

Of course, p must also be an integer so we round up and accept any overlap that occurs between adjacent planes. At this point we have defined an inclined SOC constellation.

Since the size of the swath is directly related to the L/D of the CAV, and the number of planes is directly related to both inclination and L/D, it follows that different combinations of inclination and L/D will require different numbers of CAVs for inclined SOC constellations. Generally, as L/D increases, the number of planes decreases; and as inclination increases, the number of planes also increases. Figure 8 shows the number of orbit planes for inclined constellations as a function of L/D and inclination. Figure 9 shows an inclined SOC constellation.

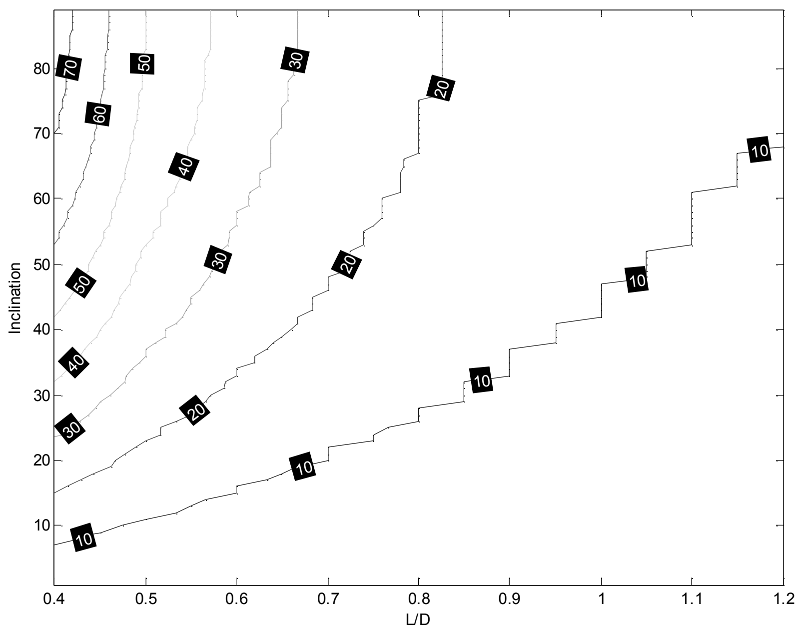


Figure 8 Number of Orbit Planes Required for Inclined SOC Constellation

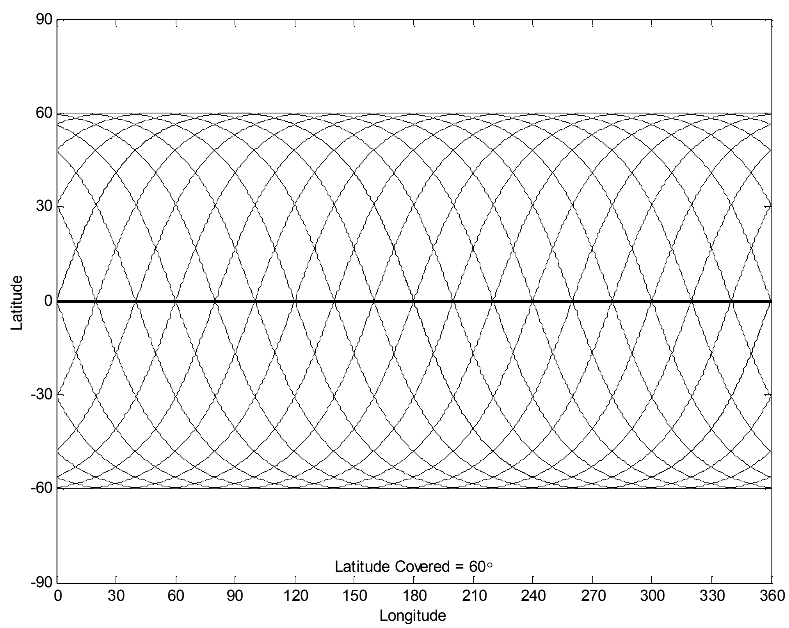


Figure 9 Inclined SOC Constellation (i = 60°, λ_{max} = 10°, p = 18)

For polar SOC constellations, Equation 6 may be simplified. This is due to the fact that at inclinations of 90°, the ascending and descending paths of the CAVs completely cover the globe. Thus, spacing the orbit planes around the entire circumference of the Earth would result in two CAVs traveling opposite directions over the same ground trace. To eliminate this redundancy, we distribute the orbit planes around only half the Earth. The number of orbit planes is given by

$$p = ceiling\left[\frac{\pi}{w}\right]. \tag{7}$$

In this case we see that as L/D increases, the number of planes decreases. Figure 10 shows the number of planes for polar constellations as a function of L/D. Figure 11 shows a polar SOC constellation.

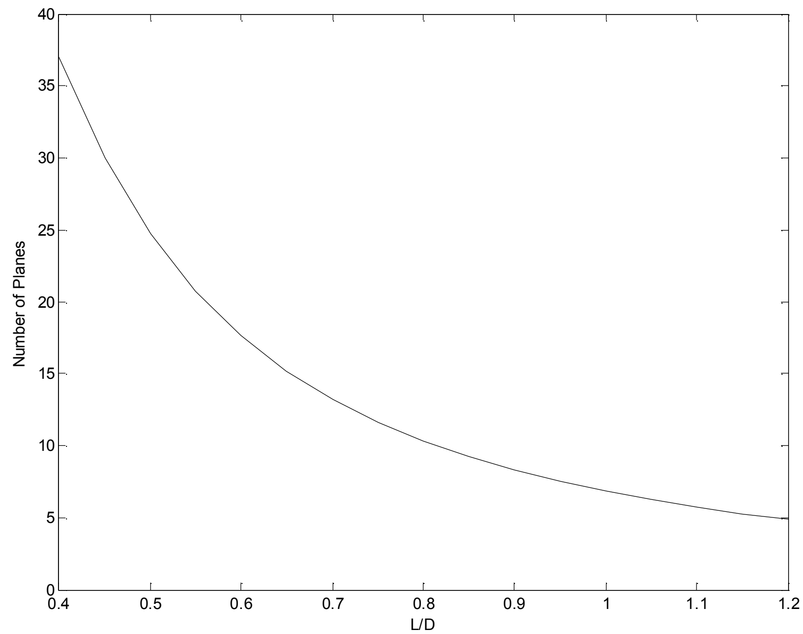


Figure 10 L/D vs Number of Planes for Polar SOC

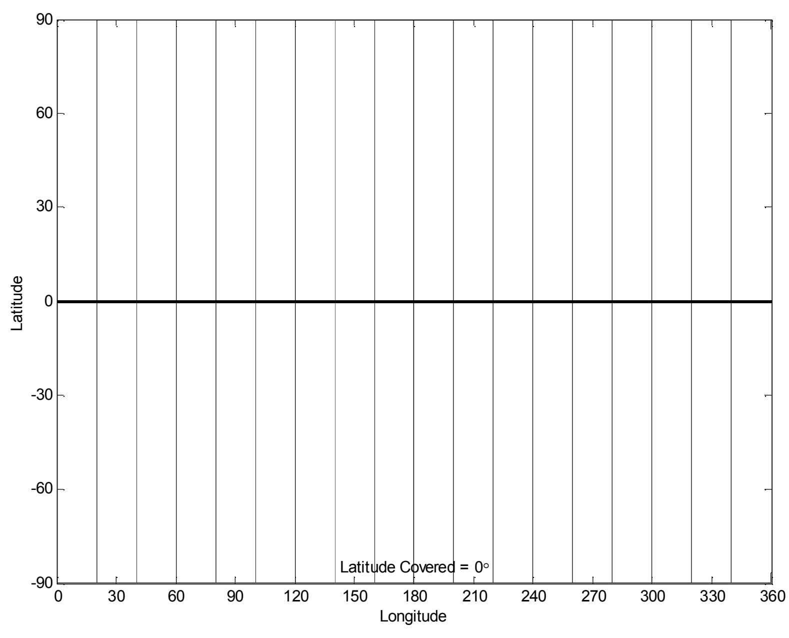


Figure 11 Polar SOC Constellation (i = 90°, λ_{max} = 10°, p = 9)

Modified Streets of Coverage Design

In Figure 9, we note that although we do indeed have complete coverage in the area of interest, we also have a great deal of redundant coverage. In the interest of economy, we might consider ways to eliminate one or more orbit planes from the inclined SOC constellation to produce a modified SOC constellation.

We first consider removing half the orbit planes, as shown in Figure 12. That is, the ascending nodes are distributed around 180°, just as in the polar SOC constellation. There are now large areas of non-coverage; in fact, the only latitude fully covered is the equator. This is obviously not an effective solution, so we next consider removing a smaller number of orbit planes. In Figure 13, only three orbit planes have been removed. Coverage is only slightly reduced and full coverage still exists nearly to the inclination of the constellation. This result is promising, and we now consider how altering the inclination of the modified constellation affects the minimum number of orbit planes required for full coverage below a specified latitude.

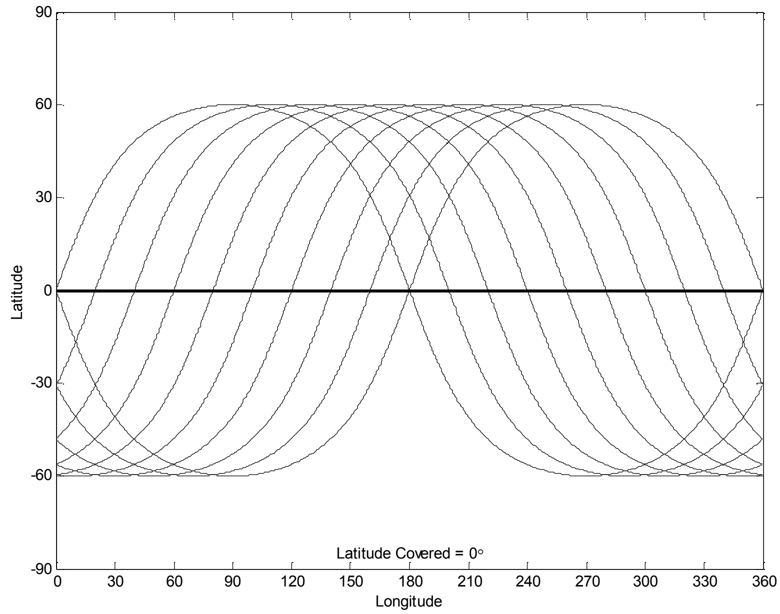


Figure 12 Modified Inclined SOC Constellation with coverage at 0° (i = 60°, λ_{max} = 10°, p = 9)

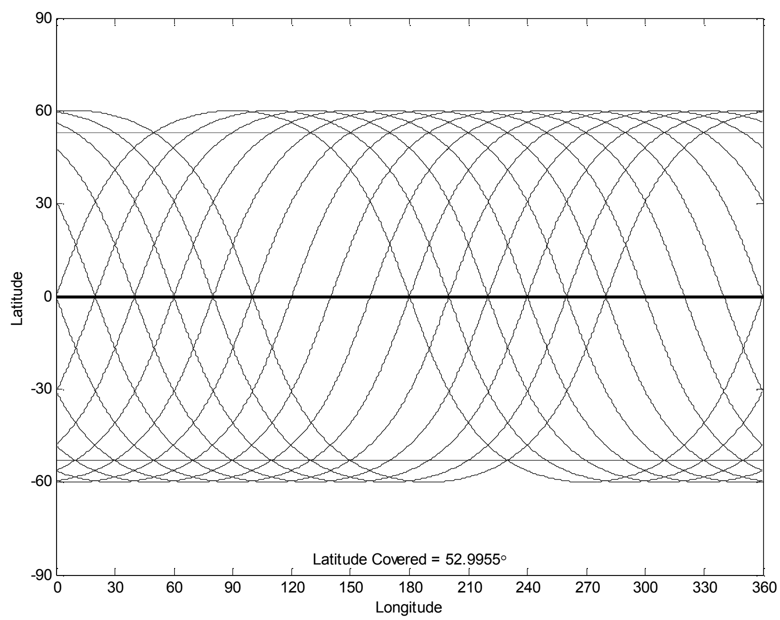


Figure 13 Modified Inclined SOC Constellation With Coverage at ~ ±52° (i = 60°, λ_{max} = 10°, p = 15)

We first investigate how far apart two planes can be while still covering a given latitude. To determine this value, we must understand how the spacing between orbit planes relates to the intersection of their swaths (5:62-3). Figure 14 illustrates the geometry involved.

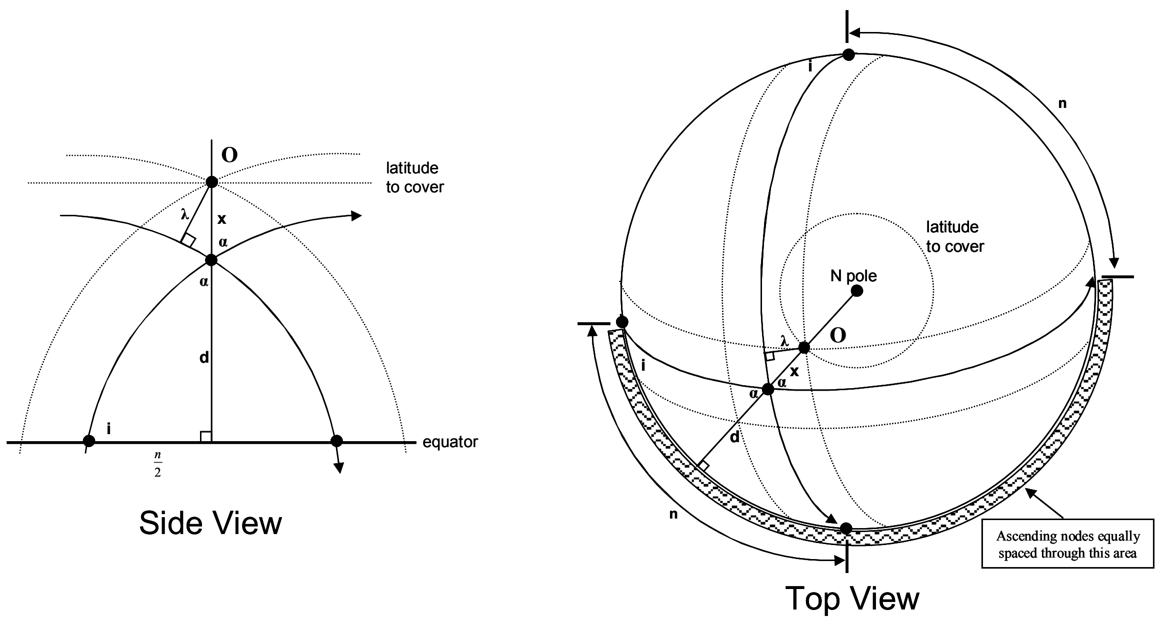


Figure 14 Swath Intersection Geometry

Several formulas from spherical trigonometry may be applied:

$$\sin\left(\frac{n}{2}\right) = \frac{\tan d}{\tan i}$$
$$\sin\alpha = \frac{\cos i}{\cos d} \qquad (8)$$
$$\sin x = \frac{\sin\lambda}{\sin\alpha}$$

We also know that

$$(d + x) = \phi_{req} \tag{9}$$

where φ_{req} is the latitude of required coverage. These relationships can be combined to eliminate α and x and produce an expression for n:

$$\sin\left(\frac{n}{2}\right) = \frac{\sin\phi_{req}\cos i - \sin\lambda}{\cos\phi_{req}\sin i} \tag{10}$$

The complete derivation of this formula is given in Appendix B.

For this application, we define φ_{req} and then find a value for inclination such that the number of orbit planes is minimized. An inclined SOC constellation in which inclination is equal to the latitude of interest serves as a baseline from which we hope to improve. We will show that it is more efficient to use orbit inclinations that are somewhat higher than φ_{req}.

To create modified SOC constellations, we must find the smallest value of n such that Equation 9 is satisfied. Stated another way, we are seeking a nodal spacing such that the intersection of the upper boundaries of two swaths occurs at φ_{req}, as shown by Point O in Figure 14. We choose a value for i and find n using Equation 10. The range over which the ascending nodes are distributed must be at least equal to π but less than 2π, meaning the additional range n must be between 0 and π. This value n is then inserted

into Equation 6 to determine the number of planes required for full coverage at the desired latitude and inclination:

$$p = ceiling\left[\frac{\pi + n}{c}\right] \qquad (11)$$

The numerator in Equation 11 reflects a nodal coverage of $\pi + n$ radians rather than 2π radians. This process is repeated for all values of i between φ_{req} and 90° to produce a solution curve unique to this particular value of φ_{req}.

The results of this process, compared to inclined SOC and polar SOC constellations, are shown in Figure 15 and Figure 16 for two different swath widths. In these examples, the curve on the far left represents the number of orbit planes required to create inclined SOC constellations over the full range of possible latitudes. Each marker along this curve indicates the number of planes required at a specific φ_{req}. The curves to the right show the number of planes and inclinations required for a modified SOC constellation covering that latitude. Each of these is marked with the corresponding symbol for its particular φ_{req}. The horizontal line represents the number of orbit planes required to create a polar SOC constellation.

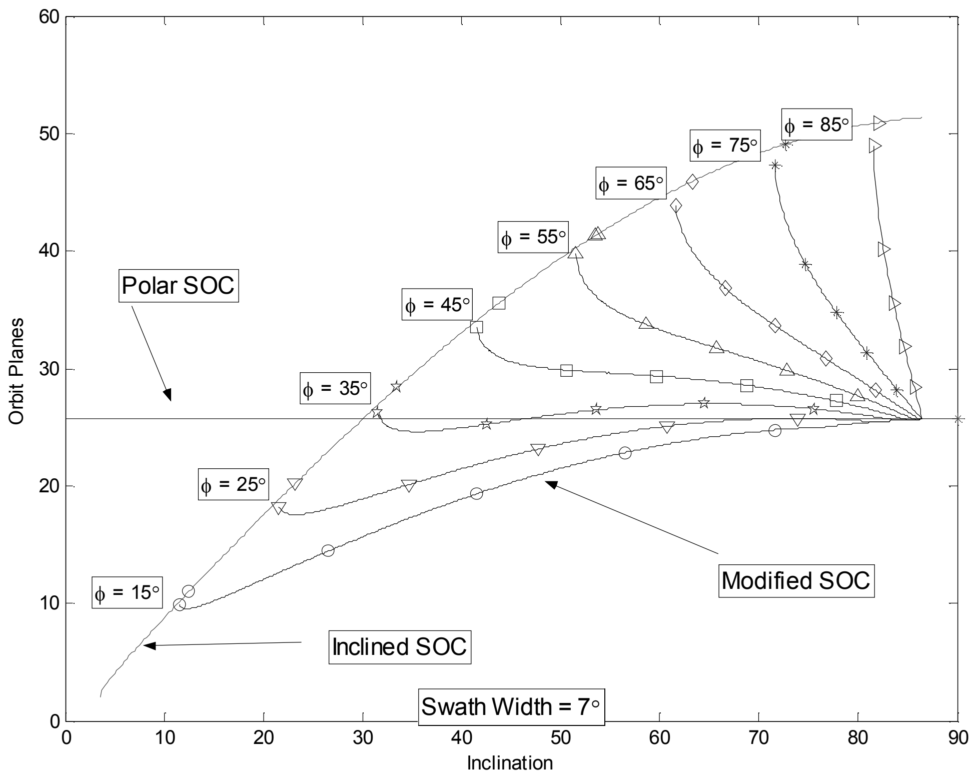


Figure 15 Orbit Planes Required at Various Latitudes for Swath Width = 7°

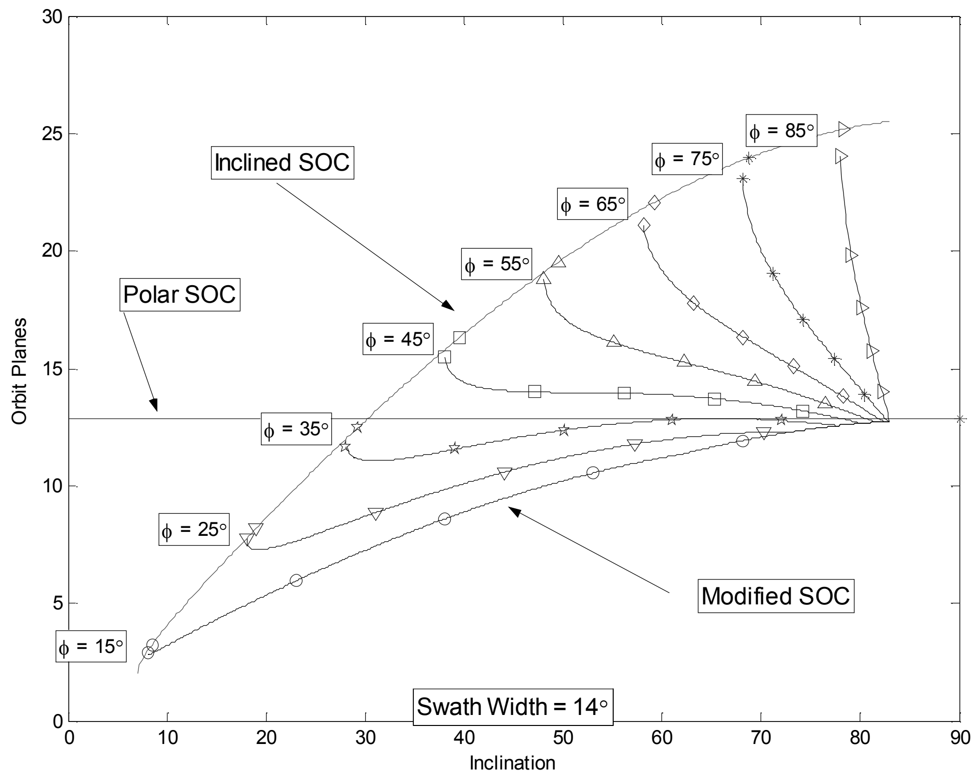


Figure 16 Orbit Planes Required at Various Latitudes for Swath Width = 14°

Numeric Constellation Design

We now employ Genetic Algorithm (GA) techniques in an attempt to further optimize the constellation. Briefly, GA allows for an examination of a large search space using techniques borrowed from the biological processes of evolution (12:207-10). Individual variables are designated as genes, and the genes are arranged into chromosomes. Each chromosome represents one particular arrangement of the problem variables (in this case representing a constellation of CAVs) which may then be manipulated by genetic operations such as mutation, crossover, and inversion. These

processes are pseudo-random and given enough time and appropriate rules for determining the fitness of each chromosome, will converge to some optimal solution (or one of a set of pareto-optimal solutions) (2:2).

Defining the Problem

The quantity to be minimized is the total number of CAVs required for 100% coverage of the latitudes of interest. A constellation is described by both the number of CAVs it contains and the coverage it provides. These two parameters are combined within a fitness function which defines the total effectiveness of the constellation.

In all GA applications the problem must be represented as a chromosome with distinct parts that can be manipulated by the GA processes. Here, we choose to encode the constellation as a fixed-length binary string. Before we can take this step, several intermediate encoding steps are necessary to ensure proper operation of the GA.

Encoding and Decoding the Chromosome

Each CAV in the constellation is fully described by its orbital elements: semi-major axis, inclination, eccentricity, right ascension of ascending node, argument of perigee, and mean anomaly. In this application, semi-major axis is fixed. Additionally, all orbits are assumed circular which means argument of perigee and mean anomaly become undefined; in this case we represent the CAV's position within the orbit using argument of latitude (9:28-31). Therefore, each CAV can be fully described using only inclination, right ascension, and argument of latitude, and any constellation of CAVs may be defined by an ($N \times 3$) table of these values.

One approach to building a chromosome from this table is to simply arrange the rows one after the other into a single vector of length 3*N*. However, there are several shortfalls with this method. First, when performing a crossover operation, two chromosomes must be cut at some point along their length. The chromosomes then exchange all information contained after the cut with each other. To retain all the information for each CAV, the cut must occur along the length of the chromosome in some multiple of three. Currently, the software used for the GA process is unable to enforce this condition, and as a result any offspring from the crossover operation are likely to be missing some information. Rather than attempt to work around these issues, we adopt a table lookup approach.

We create a table which holds all possible combinations of inclination, right ascension, and argument of latitude within certain fidelity. Each row of this table represents one possible set of values describing a single CAV. Individual genes are now reduced to a single integer representing the appropriate row in the lookup table. Figure 17 illustrates the difference between the two approaches.

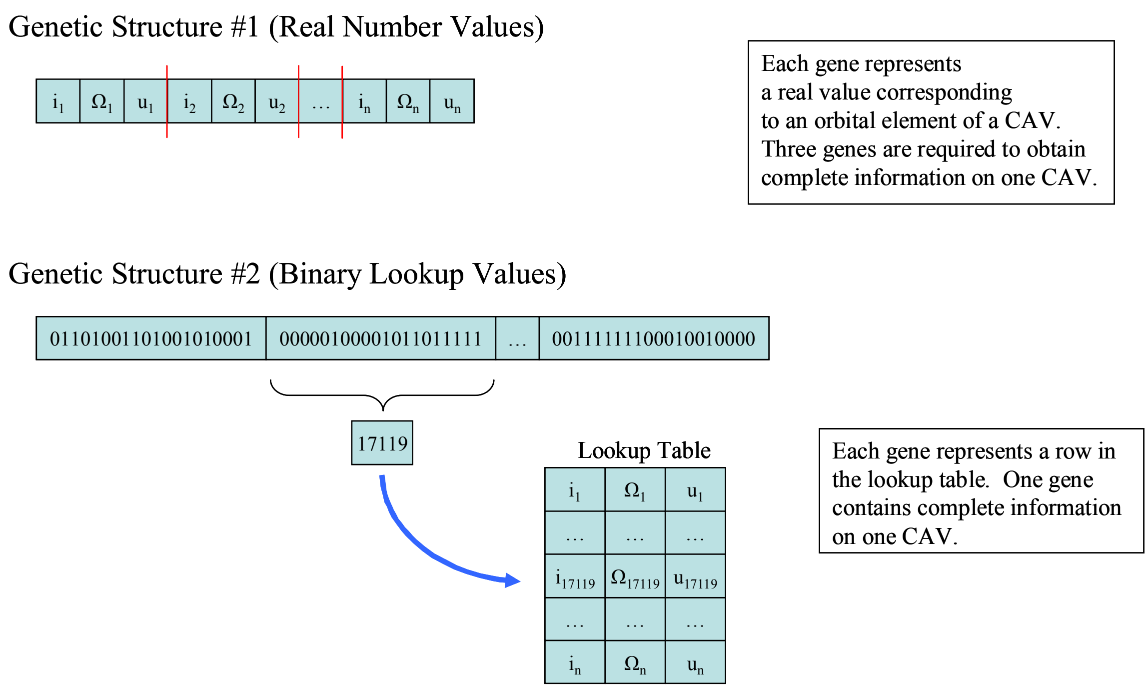


Figure 17 GA Encoding Scheme Comparison

We construct the lookup table by allowing the variables of interest to increment in steps of .05 radians; inclination varies through a range of π radians while right ascension and argument of latitude vary through 2π radians. This allows for a thorough search of possible configurations while avoiding the computational overhead of an extremely large matrix. The size of this table is given by

$$size = \left(\frac{2\pi}{.05}\right)\left(\frac{2\pi}{.05}\right)\left(\frac{\pi}{.05}\right) = 992{,}200\,. \tag{12}$$

To accommodate this many values, a 20-digit binary string, which may take on any value between 0 and 1,048,576 is ideal. Analytic results show that we can expect to

never have more than 100 total CAVs (using current *L/D* values), so we select a fixed-length chromosome of 20 × 100 = 2000 bits. This binary approach allows easy manipulation of the information in the chromosome by the genetic processes.

This method also allows the algorithm to easily remove CAVs from a constellation. There are 56,376 unused rows in the lookup table, and their contents are assigned to the null set. When the algorithm references one of these 'empty' rows during chromosome manipulation, the interpretation is that the CAV does not exist. Likewise, removing a CAV is as simple as setting its gene to a value that references an 'empty' row in the lookup table. Since the goal is to minimize the number of CAVs, having many copies of the null CAV was thought to be desirable in increasing the likelihood of removing additional CAVs during crossover operations.

Encoding the chromosome now consists of replacing each row in the constellation matrix with its corresponding integer row number from the large lookup matrix, then converting each integer into its 20-bit binary equivalent. The final step is to rearrange the now (100 × 20) matrix into a (1 × 2000) row vector for processing by the GA code. To decode processed chromosomes, we simply reverse the steps.

Genetic Processes

Once the encoded chromosomes are created, they are processed to generate variability in the design. There are three processes which occur within each generation: selection, modification, and decimation. Selection is performed by choosing one or two chromosomes, either at random or in proportion to their fitness. Modification consists of either mutation, where each bit is subject to inversion ($1 \rightarrow 0$ or $0 \rightarrow 1$) based on a set

probability; or crossover, where two chromosomes exchange a randomly determined amount of genetic material. In both cases, chromosome length is fixed. Decimation occurs at the end of each generation and is used to eliminate the least fit members from the population, and to keep the population size at a manageable level.

Coverage Evaluation

A straightforward method to evaluate constellation coverage is to distribute evenly spaced points around the equator and around each line of latitude, then check if each point is covered at any point in the simulation. However, if the same number of points are distributed along the higher latitudes as along the equator, there will be many more points in the polar regions. This will tend to skew any figures of merit toward polar coverage, which may not be desirable. We eliminate this problem by reducing the number of points along any line of latitude by

$$points_{latitude} = (points_{equator})(\cos\phi) \tag{13}$$

We also include a random starting point on each line of latitude to prevent artificial weighting of the prime meridian. As the number of grid points increases, so does the time required to compute coverage. Tests were performed with grid spacing as low as 1° and did not show any appreciable increase in accuracy over larger values when determining coverage. A grid spacing of 5° was chosen as a good balance between grid fidelity (1656 grid points) and computational efficiency.

Fitness Evaluation

To evaluate the fitness of a constellation, each CAV in the constellation is propagated through its swath. Since the swath is defined only by $\lambda_{max,}$ grid coverage at each time step can be checked with a simple dot product calculation (see Figure 6). If a grid point is covered it is flagged, and after all CAVs are propagated the total coverage is calculated. The constellation's fitness is given by

$$f = \frac{number\ of\ CAVs}{coverage^{\,q}} \tag{14}$$

where q represents a variable exponent designed to penalize incomplete constellations. This fitness value is passed back to the GA code, which then ranks the constellation, and either retains the constellation for future generations or discards it through the decimation process.

In Equation 14, the integer q in the denominator controls the rate at which the GA algorithm converges on possible solutions. Constellations with less than 100% coverage are always penalized according to the amount of coverage they have. Early in the search, we keep the penalty low, and therefore q is small. This allows a larger variety of genetic material to remain in the population. However, as the search proceeds, we must begin eliminating those constellations with less than 100% coverage. Thus, we increase the coverage penalty, and so q increases throughout the life of the GA process. Experimentation with the algorithm leads us to set $q = 3$ at the beginning of the process

and allow it to increase along a parabolic curve until the end of the process, at which time $q = 20$. The exponent is computed by

$$q = \left(\frac{17}{(total\ generations - 1)^2} \right) (current\ generation - 1)^2 + 3 \tag{15}$$

where the value of *total generations* is input by the user.

IV. Analysis and Results

Chapter Overview

This chapter details the results of both analytic and numeric analysis of the Earth coverage problem. Minimal constellations are discussed and verified in both methods, and general design conclusions are made.

Analysis

To assist in evaluating the performance of the analytic versus the numeric methods, we choose the following CAV properties: L/D = 0.7; mass = 1000 kg; frontal surface area = 10 m^2. All CAV orbits are circular with a semi-major axis of 500 km. Bank angle was calculated at 40.1° using Equation 2 . The following values were generated by the simulation: $d_{re\text{-}entry}$ = 115.94°; $t_{re\text{-}entry}$ = 2562 sec.; λ_{max} = 7°. We choose a latitude coverage band of ± 65° for the first simulation, and ±25° for the second.

Analytic Results

We now develop a nominal constellation of CAVs based on these values. Swath length equals 182.77°, calculated using Equation 3. Swath width is given by $w = 2(\lambda_{max}) = 14°$. The number of CAVs per plane, $s = 2$, is found from Equation 4. Number of orbit planes and constellation inclination varies widely with constellation type.

For the polar constellation, $p = 14$ from Equation 7, and $i \geq 83°$.

For the inclined SOC constellation, p = 12 and i = 25° for the ± 25° case; and p = 25 and i = 65° for the ±65° case. The number of planes was determined from Equation 6, and the inclination was set equal to φ_{req}.

For the modified SOC constellation, p = 8 and i = 19.5° for the ± 25° case; and p = 14 and i = 82.5° for the ±65° case. The number of planes was determined using Equation 11, and the inclination was determined from Equation 10.

We see from Figure 15 and Figure 16 that as inclination increases, more orbit planes are required to fully cover the latitude band of interest when using an inclined SOC constellation. This is because the width of the swath as it crosses the equator decreases as inclination increases. However, for the modified SOC constellations, a more complicated curve results, and this trend is actually reversed for mid to high values of φ_{req}. Although the swath width at the equator is decreasing, the number of orbit planes we can remove increases, driving the total number of orbit planes down. We also note the impact of φ_{req} on the length and slope of each curve. As expected, high latitudes can only be serviced by high inclination orbits, and the effect of increasing inclination is more pronounced.

Obviously, removing orbit planes from an inclined SOC constellation improves the efficiency of the constellation. For lower-latitude coverage, these modified SOC constellations are the most efficient way to cover the area of interest. As latitude increases past a certain value, however, polar SOC constellations become more efficient. The exact latitude where this transition occurs is a function of swath width and inclination and is not analytically tractable. With sub-polar latitude coverage

requirements, polar SOC constellations generate unneeded coverage in the polar regions, but the fact that they require only half the orbit planes as inclined full SOC constellations make up for this relative inefficiency. Therefore, a polar SOC constellation is the generally the best method for providing mid- to high-latitude coverage.

If, for some reason, polar orbits are not achievable, the modified SOC constellation still provides a significant advantage over its inclined SOC counterpart. As shown in Figure 15 and Figure 16, constellation efficiency goes down as inclination decreases, but still remains better than the full inclined SOC baseline.

An illustrative example is to consider, from Figure 15, the curve representing a φ_{req} value of 35°. In this case, the full SOC solution requires approximately 29 orbit planes, while the modified SOC solution requires approximately 25 orbit planes, at an inclination of approximately 34°. The polar SOC solution in this case requires approximately 26 orbit planes.

Numeric Results

The GA process was implemented with an initial population containing several inclined SOC constellations for inclinations at and above the latitude being studied. In order to provide the algorithm enough information to begin a valid search of the solution space, approximately 100 randomly generated constellations were also included in the population. Additionally, an 'all zeros' chromosome was added. This chromosome represents an empty constellation (by referencing the null rows of the lookup table), and allows the algorithm to remove CAVs from a constellation via the crossover operation. The algorithm was run multiple times using a variety of values for *total generations*.

Tests were performed using up to 10,000 generations with no improvement over lower values. A value of 1500 generations was chosen to give a good balance between computing time and depth of search.

In each GA run, the two best constellations and the two worst constellations were plotted for analysis. As expected, the worst constellations were random assortments of CAVs, with coverage often dropping below 50%. These were included in the output to verify the algorithm was keeping a large variety of possible solutions in the population. The best constellations were similar to the analytic solutions.

The GA algorithm produced interesting results. Numerically, the best constellations found were similar to the ones obtained analytically. Figure 18 shows a solution at $\varphi_{req} = 25°$, while Figure 19 shows a solution at $\varphi_{req} = 65°$. The $\varphi_{req} = 25°$ case was identical to the analytic solution, but in the $\varphi_{req} = 65°$ case, the GA algorithm eliminated one additional orbit plane from the analytical solution, which lowered coverage values to > 97%. This was typical of the GA results in general; in most cases, the best constellations had coverage slightly less than 100%.

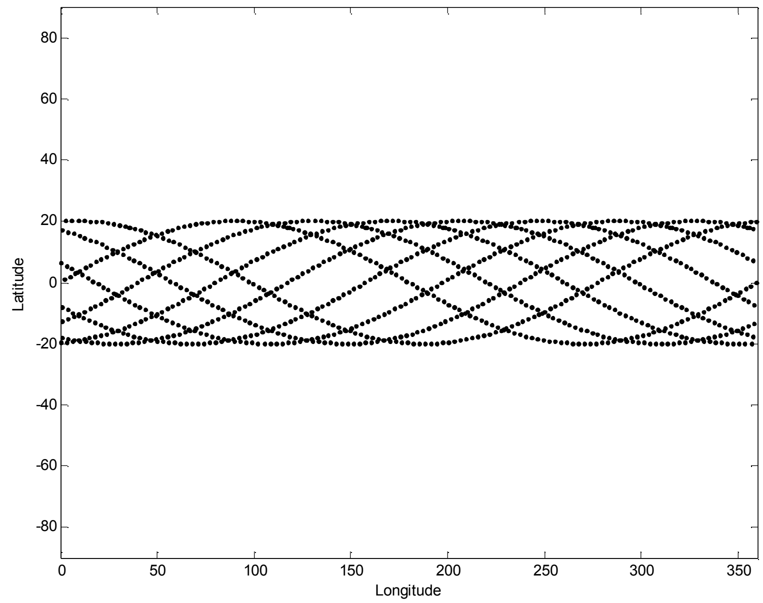


Figure 18 Example GA Result (±25° case)

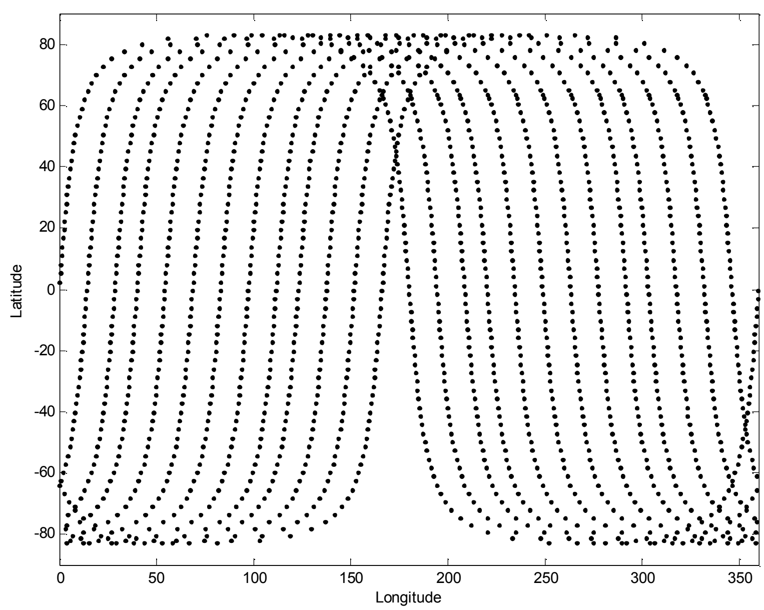


Figure 19 Example GA Result (±65° case)

Analytic vs. Numeric Comparison

At this point we wish to summarize the results of both analyses. Although the numeric analysis shows that further reductions from the analytic results are possible, it does not represent a significant savings. The general result is that polar SOC constellations are the most efficient way to cover mid to high latitudes, while modified SOC constellations become optimal at lower latitudes. The exact latitude at which this occurs is mainly a function of swath width and inclination and is not analytically tractable. Table 2 and Table 3 compare the results of the analytic and numeric analysis.

Table 2 Constellation Summary for Latitude Requirement of 25°

Constellation Type	Latitude Coverage Req't	Inclination	Number of Planes	Number of CAVs	Coverage (%)
Polar SOC	25°	≥ 83°	14	28	100
Full SOC	25°	25°	12	24	100
Modified SOC	25°	19.5°	8	16	100
GA Low	25°	19.5°	8	16	100

Table 3 Constellation Summary for Latitude Requirement of 65°

Constellation Type	Latitude Coverage Req't	Inclination	Number of Planes	Number of CAVs	Coverage (%)
Polar SOC	65°	≥ 83°	14	28	100
Full SOC	65°	65°	25	50	100
Modified SOC	65°	82.5°	14	28	100
GA High	65°	83.1°	13	26	97.1

At this point we must acknowledge that the numeric results are somewhat disappointing. This may be due to the lack of any truly new or interesting solutions or it

may be due to shortcomings in the GA routine itself; without further investigation we cannot confirm either suspicion. However, the results can be partially explained by addressing some concerns regarding the operation of the GA routine. There were several roadblocks to overcome in this formulation, not all of which were satisfactorily resolved.

First, the fidelity of the lookup table was of some concern. Although each orbital element was incremented in steps of .05 radians (equivalent to 2.86°), a finer table might lead to more robust results. The possibility exists that an optimal solution was missed due to the proper value not existing in the lookup table. However, the excessive computation time associated with a larger table prohibited its use in this study.

Second, calculation of the constellation's fitness value was of some concern. The problem of properly weighting Earth coverage was addressed early in the research by allowing the exponent q in Equation 14 to vary throughout the simulation. However, the evolution of the population is very sensitive to the value of q and the rate at which q is allowed to grow during the simulation. Without further experimentation we cannot definitively state that the fitness function ideally calculates the true fitness of the constellation.

Finally, the issue of population diversity and its effect on convergence of the algorithm must be addressed. Originally, the initial population was seeded with a single inclined SOC solution, an 'all zeros' chromosome, and approximately 100 randomly generated chromosomes. This setup produced constellations remarkably similar to the modified SOC constellations discussed above. It was reasoned that the presence of the 'zero chromosome' allowed the algorithm to remove individual CAVs in the inclined

SOC constellation. However, when a different mix of initial chromosomes was used, the routine converged to a constellation containing only one CAV. This dependence on a suitable initial population is a documented shortfall of GA searches in general (12:107). The current assortment of constellations in the initial population was generated through extensive experimentation; there may be a better mix of constellations that would yield better results.

V. Conclusions and Recommendations

Conclusions of Research

For mid to high latitude coverage requirements, polar SOC constellations are the most efficient method of providing full coverage within 90 minutes of a decision to de-orbit the CAV. For low latitude coverage requirements, or in circumstances where polar orbits are not achievable, modified SOC constellations are the most efficient method. The exact latitude at which this transition takes place cannot be obtained analytically, but is mainly a function of swath width and inclination. Furthermore, in cases where less than 100% coverage is acceptable, additional orbit planes can be removed from the modified SOC constellation. The latitude at which these modified SOC constellations become more advantageous than a polar SOC constellation varies with the swath width of the CAVs in the constellation.

Significance of Research

The elimination of one or more orbit planes from inclined SOC constellations results in launch cost savings (fewer launches required) as well as overall system cost savings (fewer total CAVs required). The design paradigm addressed here is valid for constellations which do not require continuous coverage, although the method could be extended to more standard applications. If a ring of satellites in a single orbit plane can produce a continuous swath of coverage on the ground, the method presented here may be applied to design of the constellation.

Also worth noting is the realization that reducing coverage requirements can eliminate additional orbit planes in cases where the cost per orbit plane far exceeds the value of complete coverage. A trade study using this paradigm could be conducted in the design phase of almost any constellation of this type.

Recommendations for Future Research

In the future, a more robust study of CAV re-entry should be conducted to more accurately and completely define the footprint and swath size of the vehicle. Optimization of the re-entry trajectory could yield greater swath size and a subsequent reduction in constellation size. Creating an algorithm to model the irregular shape of the footprint, and thus the entire swath, would further maximize the potential of a single CAV and lead to additional reductions in constellation size.

Adding a delta-v capability while on orbit would also change the CAV's footprint and swath size, with a reduction in the number of CAVs required being a likely outcome. The simplified analysis presented here is not sufficient to model the ability of the CAV to change its orbit plane before re-entry. Significant further study is necessary to include this capability.

The fidelity of the study would be improved by adding Earth rotation, J2, and other perturbations into the model. These effects will not change the overall shape of the constellations, but will provide further validation of the concept as well as a logical link to the next part of the study.

Follow-on research should attempt to model a complete system of CAVs along with their timing and target opportunities. Some specifics would entail creating an

algorithm to define which CAV to select for deployment against a specific target, given the current time and time-on-target information. The study would need to include a robust model for orbital motion, a complete description of re-entry times to every part of the footprint, and a method for choosing the CAV most likely to arrive at the target within the 90 minute time constraint.

Finally, a more robust and reliable GA routine should be implemented. Many of the shortfalls of the GA routine were addressed in the analysis section of this paper and will need to be addressed before the GA search can be considered complete. Although further reductions in constellation size may not be realistic, this endeavor should be undertaken to eliminate any doubt on the matter.

Appendix A

Notation

EOM VARIABLES

$h = altitude$

$\theta = Earth\ longitude$

$\phi = Earth\ latitude$

$v = Earth - relative\ velocity$

$\gamma = flight\ path\ angle, measured\ downward\ from\ local\ horizontal$

$\Psi = heading\ angle, measured\ from\ local\ latitude\ to\ the\ projection\ of\ v$
$\quad onto\ the\ local\ horizontal$

$\sigma = bank\ angle, measured\ from\ local\ vertical\ to\ the\ lift\ vector$

$$\frac{D}{m} = \frac{\rho_0 e^{\left(\frac{-h}{H}\right)} v^2}{2\beta_m}$$

$$\frac{L}{m} = \frac{\rho_0 e^{\left(\frac{-h}{H}\right)} v^2}{2\beta_m} \left(\frac{c_l}{c_d}\right)$$

$$\beta_m = \frac{m}{c_d S}$$

$R_E = Earth\ radius$

$H = scale\ height$

$\rho_0 = atmospheric\ density\ at\ sea\ level$

$S = surface\ area\ of\ reentry\ vehicle\ normal\ to\ velocity\ vector$

OTHER VARIABLES

$a = semi - major\ axis$

$\omega = orbital\ mean\ motion$

$i = inclination$

$\Omega = right\ ascension\ of\ ascending\ node$

$u = argument\ of\ latitude$

$\lambda_{max} = maximum\ crossrange\ capability$

$w = swath\ width$

$l = swath\ length$

$c = swath\ width\ at\ equator\ crossing\ for\ given\ inclination$

$s = number\ of\ satellites\ per\ orbit\ plane$

$p = number\ of\ orbit\ planes$

$d = latitude\ of\ ground\ trace\ crossing$

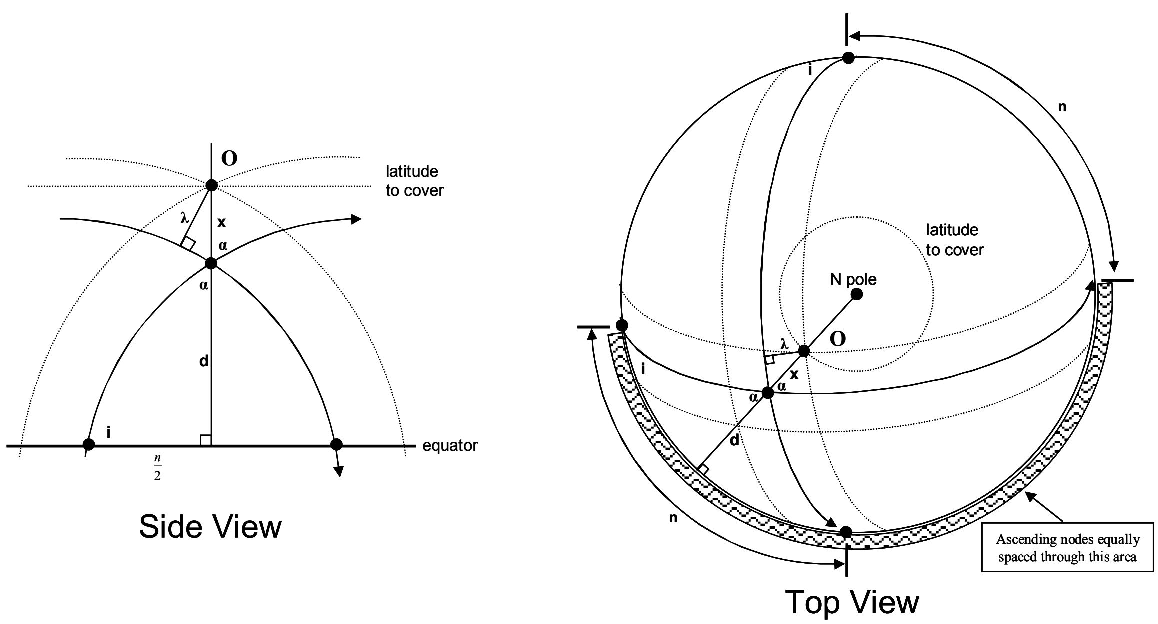
$x = latitude\ of\ upper\ swath\ boundaries\ crossing$

$n = ascending\ node\ spacing\ between\ orbit\ plane\ at\ \Omega = 180° \ and\ last\ orbit\ plane\ used$

Appendix B

Nodal Spacing Derivation

We are seeking an analytic expression for the value *n* in Equation 10 and Equation 11 which is used to determine the range of nodal crossings in a modified SOC constellation. This derivation is based on the work of Rider (5:62). In Rider's work, *n* is known and λ is the quantity being sought. Our approach differs from the reference that λ is known and *n* is unknown; and that *n* is defined differently.



We reproduce Figure 14 here and begin with the following relations from spherical trigonometry:

$$\sin\left(\frac{n}{2}\right) = \frac{\tan d}{\tan i} \tag{A1}$$

$$\sin \alpha = \frac{\cos i}{\cos d} \tag{A2}$$

$$\sin x = \frac{\sin \lambda}{\sin \alpha} \tag{A3}$$

We also make use of the trigonometric identity

$$\sin(a+b) = \sin a \cos b - \cos a \sin b \tag{A4}$$

and note from Equation 9 that

$$x = \phi_{req} - d \tag{A5}$$

Substitution of Equation A5 into Equation A3 yields

$$\sin(\phi_{req} - d) = \frac{\sin \lambda}{\sin \alpha} \tag{A6}$$

Inserting Equation A2 and applying the identity in Equation A4, we obtain

$$\frac{\sin \lambda \cos d}{\cos i} = \sin \phi_{req} \cos d - \cos \phi_{req} \sin d \tag{A7}$$

This expression can be rearranged to produce

$$\tan d = \tan \phi_{req} - \frac{\sin \lambda}{\cos i \cos \phi_{req}} \tag{A8}$$

Substituting Equation A8 back into Equation A1 and rearranging terms gives

$$\sin\left(\frac{n}{2}\right) = \frac{\sin \phi_{req} \cos i - \sin \lambda}{\cos \phi_{req} \sin i} \tag{A9}$$

This relationship gives n entirely in terms of the known values φ_{req}, λ, and i.

Appendix C

MATLAB© Code

Re-entry Simulation

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% REENTRY SIMULATION.  CALLS EOMS FILE FOR INTEGRATION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [lambda,maxtheta,maxtime1] = simple_reentry_reference(LDVALUE,altitude);
global cd0 cl beta betam sigma lambda maxtime1
global target
global h0 theta0 phi0 v0 gam0 psi0 maxtheta maxphi
global G0 SCALE RHO0 MASS FR_AREA RE

% Times
tstep = 1;  %sec
tfinal = 4000;  %sec

% Earth and atmospheric parameters
RE = 6378000; G0 = 9.8;
RHO0 = 1.752; SCALE = 6700;

% Vehicle parameters
cd0 = 1; cl = LDVALUE;
beta = (MASS*G0)/(cd0*FR_AREA); betam = beta/G0;
sigma = sigmaopt(cl,cd0);

% Initial Conditions
h0 = altitude*1000; theta0 = 0; phi0 = 0;
v0 = sqrt(RE*G0) - 130; gam0 = 0; psi0 = 0;

% Generate reference trajectory with sigma optimal
sigma = sigmaopt(cl,cd0);
y0 = [h0 theta0 phi0 v0 gam0 psi0];
options = odeset('RelTol',1e-08,'AbsTol',1e-10*ones(1,6));
[t,y] = ODE45('simple_reentry_eoms',0:tstep:tfinal,y0,options);

for i = 1:tfinal;
   if y(i,1) > 0
      p(i,1) = y(i,1);
      p(i,2) = y(i,2);
      p(i,3) = y(i,3);
```

```
        p(i,4) = y(i,4);
        p(i,5) = y(i,5);
        p(i,6) = y(i,6);
    else break
    end
end

[maxphi,maxtime1] = max(p(:,3));
lambda = maxphi;
maxphi = maxphi*(180/pi);
maxtheta = p(maxtime1,2)*(180/pi);
target = maxtheta + maxphi;
missvalue = target - maxtheta;

% Now try to achieve theta = target by using differing sigma values and
% roll reversals.  If the SMV is short, decrease sigma; if long, increase
% sigma.
sigma = sigmaopt(cl,cd0);
q = p;
while missvalue > .1;
    clear q;
    y0 = [h0 theta0 phi0 v0 gam0 psi0];
    options = odeset('RelTol',1e-08,'AbsTol',1e-10*ones(1,6));
    [t,y] = ODE45('reentry2eoms_nolift',0:tstep:tfinal,y0,options);
    for i = 1:tfinal;
        if y(i,1) > 0
            q(i,1) = y(i,1);
            q(i,2) = y(i,2);
            q(i,3) = y(i,3);
            q(i,4) = y(i,4);
            q(i,5) = y(i,5);
            q(i,6) = y(i,6);
        else break
        end
    end
    [maxtheta2,maxtime2] = max(q(:,2));
    maxtheta2 = maxtheta2*(180/pi);
    missvalue = target - maxtheta2;
    sigmacorr = missvalue;
    sigma = sigma - sigmacorr*(pi/180);
end
maxtime1
maxtime2
timediff = abs(maxtime1 - maxtime2)
```

```
% Pass maximum time back to main program for use in coverage statistics
if maxtime1 > maxtime2;
   downrange_time = (maxtheta*(pi/180))/m_motion;
else
   downrange_time = maxtime2;
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% REENTRY EQUATIONS OF MOTION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ydot] = simple_reentry_eoms(t,y)

global cd0 cl betam sigma
global RE G0 SCALE RHO0
global h0 theta0 phi0 v0 gam0 psi0 turn0

if y(6,1) >= psi0 + turn0;
   sigma = 0;
end

%dh/dt
ydot(1,1) = -y(4)*sin(y(5));

%dtheta/dt
ydot(2,1) = (y(4)*cos(y(5))*cos(y(6)))/(cos(y(3))*(RE + y(1)));

%dphi/dt
ydot(3,1) = (y(4)*cos(y(5))*sin(y(6)))/(RE + y(1));

%dv/dt
ydot(4,1) = -(RHO0*exp(-y(1)/SCALE)/(2*betam))*y(4)^2 + G0*sin(y(5));

%dgamma/dt
ydot(5,1) = -(1/y(4))*((y(4)^2/(RE + y(1)))*cos(y(5)) + ((RHO0*
exp(-y(1)/SCALE)/(2*betam))*y(4)^2*(cl/cd0))*cos(sigma) - G0*cos(y(5)));

%dpsi/dt
ydot(6,1) = (1/(y(4)*cos(y(5))))*((y(4)^2/(RE + y(1)))*(cos(y(5))^2)*sin(y(6))*tan(y(3))
+ ((RHO0*exp(-y(1)/SCALE)/(2*betam))*y(4)^2*(cl/cd0))*sin(sigma));
```

Earth Grid

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% THIS PROGRAM CREATES A GRID OF POINTS SPACED EQUALLY AROUND
% THE EARTH. IT ACCOUNTS FOR BUNCHING AT THE POLES AS WELL AS
% RANDOMIZING THE START POSITION OF THE FIRST POINT ALONG EACH
% LINE OF LATITUDE.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global tol num lat long earth_grid grid_tolerance grid_points
global x1 y1 z1 max_lat

% User sets the spacing between grid points, entered in degrees.
tol = grid_tolerance*(pi/180);
% Initialize to zero degrees in both lat and long.
% Also introduce randomness into longitude start so prime meridian doesn't
% get too much weight.
lat = 0;
long = 0 + rand*tol;
num = 0;
a = 1;

% Outer loop increments latitude and keeps longitude starting point random.
while lat <= max_lat;
   % Inner loop increments longitude
   while long <= 2*pi;
      earth_grid(a,1) = lat*(180/pi);
      earth_grid(a,2) = long*(180/pi);
      earth_grid(a,3) = 0;
      % This block takes care of Southern Hemisphere by mirroring points
      % when latitude is not zero.
      if lat > 0
         a = a + 1;
         earth_grid(a,1) = -lat*(180/pi);
         earth_grid(a,2) = long*(180/pi);
         earth_grid(a,3) = 0;
      end
      num = (2*pi/tol)*cos(lat);
      long = long + (2*pi/num);
      if long <= 2*pi
         a = a + 1;
      end
   end
   long = 0 + rand*tol;
   lat = lat + tol;
```

```
    a = a + 1;
end
[grid_points,r] = size(earth_grid);

grid_indx = [1:grid_points];
current_lat = pi/2 - (earth_grid(grid_indx,1).*(pi/180));
current_long = earth_grid(grid_indx,2).*(pi/180);
x1 = cos(current_long).*sin(current_lat);
y1 = sin(current_long).*sin(current_lat);
z1 = cos(current_lat);
```

Constellation Function

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CREATES A NOMINAL CONSTELLATION BASED ON INPUTS FROM THE
% REENTRY SIMULATION.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [constellation,num_planes,num_sats] =
          constellation_func2(lambda,maxtime1,inc,m_motion,n)

global constellation earth_coverage TTT w

% Determine number of SMVs for this value of lambda
smv_per_plane = ceil(2*pi/mod(m_motion*(TTT - maxtime1),2*pi));
if inc + lambda >= pi/2;
    num_planes = ceil(pi/w);
else
    num_planes = ceil((pi + n)/w);
end
num_sats = num_planes*smv_per_plane;
raan_init = 0;
arg_lat = 0;
count = 1;
u_incr = 2*pi/smv_per_plane;
raan_incr = w;
constellation = zeros(num_sats,3);
while count < num_sats;
    for count2 = count:count + smv_per_plane - 1;
        constellation(count2,1) = inc;
        constellation(count2,2) = raan_init;
        constellation(count2,3) = arg_lat;
        arg_lat = mod(arg_lat + u_incr,2*pi);
    end
```

```
    count = count2 + 1;
    raan_init = raan_init + raan_incr;
    arg_lat = 0;
end
```

Genetic Algorithm

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% THIS IS THE MAIN PROGRAM IN THE GENETIC SEARCH.  IT HAS FOUR
%       PARTS:
% 1) OBTAIN NOMINAL REENTRY PARAMETERS BASED ON SMV INPUTS
% 2) BUILD AN INITIAL POPULATION OF CONSTELLATIONS
% 3) GA SEARCH TO FIND OPTIMAL CONSTELLATION
% 4) DISPLAY RESULTS OF TWO BEST AND TWO WORST CONSTELLATIONS
%       GRAPHICALLY
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all; pack;
global MASS LDVALUE FR_AREA altitude earth_grid lambda TTT big_array bigrows
global grid_tolerance sma ecc inc m_motion MU num_sats fitness A fitfun h k
global h0 theta0 phi0 v0 gam0 psi0 RE G0 RHO0 SCALE max_lat maxgen slope yint
global planes_mod w

% Inputs for the SMV and the orbit, as well as the tolerances for the Earth
% grid and latitude limits
MASS = 1000;                % kg
LDVALUE = .7;               % lift/drag
FR_AREA = 10;               % m^2
deltav = 0;                 % m/s
altitude = 500;             % km
max_lat = 65*(pi/180);      % deg
grid_tolerance = 5;         % deg
MU = 3.986e5;               % constant
TTT = 5400;                 % sec

% Nominal orbital elements
sma = 6378 + altitude;      % km
ecc = 0;                    % no units

% Calculate orbital period
m_motion = sqrt(MU/sma^3);  % rad/s
period = 2*pi/m_motion;     % sec

% Get displacement distance from reference reentry profile
```

```
[lambda,maxtheta,maxtime1] = simple_reentry_reference(LDVALUE,altitude);

% Set up and calculate Earth grid
earthgridpoints

% Generate the full constellation array for all possible SMVs
binsize = 20;
big_array = zeros(2^binsize,4);
inc_int = .05;
raan_int = .05;
arglat_int = .05;
values = const_perms(inc_int,raan_int,arglat_int);
values(:,4) = 1;
[valrows,valcols] = size(values);
big_array(1:valrows,1:4) = values;
[bigrows,bigcols] = size(big_array);

% Setup max generations and fitness function exponent

maxgen = 5000;
maxexp = 20;
h = 1; k = 3;
fitfun = 3;
% Linear
if fitfun == 1;
   slope = (maxexp - k)/(maxgen - h);
   yint = 1 - slope;
end
% Right parabola
if fitfun == 2;
   A = (maxgen - h)/((maxexp - k)^2);
end
% Up parabola
if fitfun == 3;
   A = (maxexp - k)/((maxgen - h)^2);
end

% Initialize Genetic Search

seed = 601387;
gs_init(seed);

rand('state', 91403)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% DEFINE THE INITIAL POPULATION BASED ON MAX_LAT.  INCLUDE
% HIGHER% INCLINATIONS EVENLY SPACED FROM MAX_LAT TO 90DEG.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Define the range of inclinations the constellation function will consider
mem_count = 1;
% Define the baseline SOC constellation
count2 = 1;
for inc = [lambda:.002:pi/2 - lambda];
    w = 2*asin(sin(lambda)/sin(inc));
    for n = [0:.002:pi];
        d = atan(sin(n/2)*tan(inc));
        x = asin((sin(lambda)*cos(d))/cos(inc));
        if d + x >= max_lat;
            nplanes = (pi + n)/w;
            if nplanes >= 1;
                planes3(count2) = (pi + n)/w;
                i3(count2) = inc;
                n3(count2) = n;
                count2 = count2 + 1;
            end
            break
        end
    end
end
[planes_mod,p_indx] = min(planes3);
m_inc = i3(p_indx);
m_n = n3(p_indx);
[full_1,mod_1,full_sats,mod_sats] =
constellation_func_mod(lambda,maxtime1,m_inc,m_motion,m_n);

% Assign index numbers from big_array to each CAV in each constellation
% and pad the remainder of the chromosome with zeros
base_vect = zeros(1,100);
[full_1rows,full_1cols] = size(full_1);
for aa = 1:full_1rows;
    aarow = find(big_array(:,1) <= full_1(aa,1) + inc_int/2 & big_array(:,1) >=
            full_1(aa,1) - inc_int/2  & big_array(:,2) <= full_1(aa,2) + raan_int/2 &
            big_array(:,2) >= full_1(aa,2) - raan_int/2 & big_array(:,3) <= full_1(aa,3) +
            arglat_int/2 & big_array(:,3) >= full_1(aa,3) - arglat_int/2);
    base_vect(aa) = aarow;
end
full_1chrom = gs_blank(reshape(dec2bin(base_vect,binsize)',1,100*binsize));
mem_id = ['mem_id' num2str(mem_count)];
mem_id = gs_new('Pop1',full_1chrom)
```

```
mem_count = mem_count + 1;

base_vect = zeros(1,100);
[mod_1rows,mod_1cols] = size(mod_1);
for aa = 1:mod_1rows;
   aarow = find(big_array(:,1) <= mod_1(aa,1) + inc_int/2 & big_array(:,1) >=
         mod_1(aa,1) - inc_int/2 & big_array(:,2) <= mod_1(aa,2) + raan_int/2 &
         big_array(:,2) >= mod_1(aa,2) - raan_int/2 & big_array(:,3) <= mod_1(aa,3) +
         arglat_int/2 & big_array(:,3) >= mod_1(aa,3) - arglat_int/2);
   base_vect(aa) = aarow;
end
mod_1chrom = gs_blank(reshape(dec2bin(base_vect,binsize)',1,100*binsize));
mem_id = ['mem_id' num2str(mem_count)];
mem_id = gs_new('Pop1',mod_1chrom)
mem_count = mem_count + 1;

% Now add constellations from a range of inclinations
inc_fidelity = (i3(end) - i3(1))/length(i3);
inc_increment = 20;
inc_spacing = (pi/2 - min(i3))/inc_increment;
inc_counter = 0;
next_indx = [];
while inc_counter < inc_increment;
   while isempty(next_indx) == 1;
      next_indx = find(i3(1,:) <= (i3(1) + inc_spacing*inc_counter) + inc_fidelity &...
         i3(1,:) >= (i3(1) + inc_spacing*inc_counter) - inc_fidelity);
      inc_fidelity = inc_fidelity + .001;
   end
   next_inc = i3(next_indx(1));
   planes_mod = planes3(next_indx(1));
   next_n = n3(next_indx(1));
   [full_1,mod_1,full_sats,mod_sats] =
constellation_func_mod(lambda,maxtime1,next_inc,m_motion,next_n);

   % Assign index numbers from big_array to each CAV in each constellation
   % and pad the remainder of the chromosome with zeros
   base_vect = zeros(1,100);
   [full_1rows,full_1cols] = size(full_1);
   for aa = 1:full_1rows;
      aarow = find(big_array(:,1) <= full_1(aa,1) + inc_int/2 & big_array(:,1) >=
         full_1(aa,1) - inc_int/2  & big_array(:,2) <= full_1(aa,2) + raan_int/2 &
         big_array(:,2) >= full_1(aa,2) - raan_int/2 & big_array(:,3) <= full_1(aa,3) +
         arglat_int/2 & big_array(:,3) >= full_1(aa,3) - arglat_int/2);
      base_vect(aa) = aarow;
   end
```

```
    full_1chrom = gs_blank(reshape(dec2bin(base_vect,binsize)',1,100*binsize));
    mem_id = ['mem_id' num2str(mem_count)];
    mem_id = gs_new('Pop1',full_1chrom)
    mem_count = mem_count + 1;

    base_vect = zeros(1,100);
    [mod_1rows,mod_1cols] = size(mod_1);
    for aa = 1:mod_1rows;
        aarow = find(big_array(:,1) <= mod_1(aa,1) + inc_int/2 & big_array(:,1) >=
            mod_1(aa,1) - inc_int/2  & big_array(:,2) <= mod_1(aa,2) + raan_int/2 &
            big_array(:,2) >= mod_1(aa,2) - raan_int/2   & big_array(:,3) <= mod_1(aa,3) +
            arglat_int/2 & big_array(:,3) >= mod_1(aa,3) - arglat_int/2);
        base_vect(aa) = aarow;
    end
    mod_1chrom = gs_blank(reshape(dec2bin(base_vect,binsize)',1,100*binsize));
    mem_id = ['mem_id' num2str(mem_count)];
    mem_id = gs_new('Pop1',mod_1chrom)
    mem_count = mem_count + 1;

    % Increment latitude loop and return to top of section
    inc_counter = inc_counter + 1;
    inc_fidelity = (i3(end) - i3(1))/length(i3);
    next_indx = [];
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CREATE RANDOM MEMBERS FROM BIG_ARRAY
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

seed_array = randperm(bigrows);
linecount = 1;
for jj = mem_count + 1:mem_count + 101;
    randlength = floor(rand*100);
    next_chrom = zeros(1,100);
    next_chrom(1,1:1 + randlength) = seed_array(1,linecount:linecount + randlength);
    next_chrom = gs_blank(reshape(dec2bin(next_chrom,binsize)',1,100*binsize));
    linecount = linecount + randlength;
    IDstr = ['mem_id' num2str(jj)];
    IDstr = gs_new('Pop1',next_chrom);
end

% Throw in some "all zeros" chromosomes for variety...
zero_chrom = gs_blank(num2str(zeros(1,2000)));
for kk = 1:100;
    IDstr = ['mem_id' num2str(jj + kk)];
```

```
   IDstr = gs_new('Pop1',zero_chrom);
end

% Add some "all ones" chromosomes for even more variety...
ones_chrom = gs_blank(num2str(ones(1,2000)));
for ll = 1:100;
   IDstr = ['mem_id' num2str(jj + kk + ll)];
   IDstr = gs_new('Pop1',ones_chrom);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MAIN GENETIC MANIPULATION LOOP
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Launch interrupt buttons

gs_open_cbox;

% Evaluate initial population

mem_count = gs_popsize('Pop1');

for id = 1:mem_count
        chr1 = gs_get('Pop1', id);
        fitness = main_fitness_func_bin(chr1,3)
        mem_id = gs_set_fit('Pop1', id, fitness);


        % Check for suspend and break signals

        gs_break;
        gs_suspend;

end

% Find best member in the initial population

disp('The best member in the initial population is');
mem_ids = gs_sel_lofit('Pop1');
chr1 = gs_get('Pop1', mem_ids(1))
disp('Its fitness is')
fitness = gs_get_fit('Pop1', mem_ids(1))
best_fit_start = fitness;

%Save the best initial constellation for comparison later...
```

```
[ee] = gs_sel_lofit('Pop1');
[fitness,earth_coverage] = main_fitness_func_bin(gs_get('Pop1',ee(1)),maxgen)
aa = bin2dec(reshape(gs_unblank(gs_get('Pop1',ee(2))),20,100)')';
aa = aa';
lopop_initial = big_array(aa(find(aa),:),:);
display_initial = lopop_initial(:,1:3)*(180/pi)
pause;
% Check for break signal

gs_break;

% Genetic Search Loop

for gen = 1:maxgen
	gen

	% Trim population if over 500 members

	mem_count = gs_popsize('Pop1');
	if mem_count > 500
		mem_ids = gs_selr_hifit('Pop1');
		gs_del('Pop1',mem_ids(1));
		if mem_count-1 > 500
			gs_del('Pop1',mem_ids(2));
		end
	end

	% Select genetic operation

	op_name = gs_sel_op({'mutbin', 'xovr2'}, [0.200000, 0.800000]);

	% Select members for genetic operation

	switch char(op_name)
	case 'mutbin'
		mem_ids = gs_selr_lofit('Pop1');
	case 'xovr2'
		mem_ids = gs_selr('Pop1');
	end

	% Implement genetic operation

	off_ids = gs_op('Pop1', op_name, mem_ids(1), mem_ids(2), 0.200000);
```

```
        % Evaluate the fitness of the offspring

        for off = 1:length(off_ids)
                chr1 = gs_get('Pop1',off_ids(off));
                fitness = main_fitness_func_bin(chr1,gen)
                mem_id = gs_set_fit('Pop1', off_ids(off), fitness);

        end

        % Check for suspend and break signals

        gs_break;
        gs_suspend;

end

% Find the best individual

disp('The member with the best fitness is')
mem_ids = gs_sel_lofit('Pop1');
chr1 = gs_get('Pop1', mem_ids(1))
disp('Its fitness is');
fitness = gs_get_fit('Pop1', mem_ids(1))
best_fit_end = fitness;

% Close interrupt buttons

gs_close_cbox

% Show how much GA was able to improve over the intial population
improvement = best_fit_start - best_fit_end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% THIS SECTION PRINTS A SINGLE CONSTELLATION ALONG WITH THE
% ORBITAL ELEMENTS OF THE CAVS WITHIN IT.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[ee] = gs_sel_lofit('Pop1');
[fitness,earth_coverage] = main_fitness_func_bin(gs_get('Pop1',ee(1)),maxgen)
aa = bin2dec(reshape(gs_unblank(gs_get('Pop1',ee(1))),20,100)')';
aa = aa';
lopop = big_array(aa(find(aa),:),:);
[losats,nothing] = size(lopop);
const_elements_final = lopop(:,1:3).*(180/pi)
for cur_sat = 1:losats;
```

```
   % Get orbital elements for next SMV
   inc_now = lopop(cur_sat,1);
   raan_now = lopop(cur_sat,2);
   arglat_now = lopop(cur_sat,3);
   % Calculate values outside loop to improve speed
   cos_inc = cos(inc_now);
   sin_inc = sin(inc_now);
   count = 1;
   % Increment footprint until 90 minute limit
      for tt = arglat_now + maxtheta*(pi/180):.05:arglat_now + m_motion*(TTT -
maxtime1)+ maxtheta*(pi/180);
         % Update the SSP for this time step based on SMV's orbital elements
         SSP_lat =  asin(sin_inc*sin(tt));
         SSP_long = mod(atan(cos_inc*tan(tt)) + raan_now,2*pi);
         if mod(tt,2*pi) > pi/2 && mod(tt,2*pi) < 3*pi/2;
            SSP_long = mod(SSP_long - pi,2*pi);
         end
         SSP(count,1) = SSP_long*(180/pi);
         SSP(count,2) = SSP_lat*(180/pi);
         count = count + 1;
      end
   hold on;
   figure(1); plot(SSP(:,1),SSP(:,2),'k.')
   str1 = ['SMVs= ' num2str(losats) ', Coverage = ' num2str(earth_coverage) ...
         ', L/D= ' num2str(LDVALUE) ', \phi= ' num2str(max_lat*180/pi) ...
         ', inc= ' num2str(inc_now*(180/pi)) '\circ'];
   axis([0 360 -90 90]); grid off; box on;
   xlabel('Longitude'); ylabel('Latitude');
   text(180,-85,str1,'HorizontalAlignment','center','BackgroundColor','w','EdgeColor','k');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% THIS SECTION PRINTS THE BEST INITIAL CONSTELLATION ALONG WITH
% THE ORBITAL ELEMENTS OF THE CAVS WITHIN IT.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[losats,nothing] = size(lopop_initial);
const_elements_initial = lopop_initial(:,1:3).*(180/pi)
for cur_sat = 1:losats;
   % Get orbital elements for next SMV
   inc_now = lopop_initial(cur_sat,1);
   raan_now = lopop_initial(cur_sat,2);
   arglat_now = lopop_initial(cur_sat,3);
   % Calculate values outside loop to improve speed
   cos_inc = cos(inc_now);
```

```
    sin_inc = sin(inc_now);
    count = 1;
    % Increment footprint until 90 minute limit
        for tt = arglat_now + maxtheta*(pi/180):.05:arglat_now + m_motion*(TTT -
maxtime1)+ maxtheta*(pi/180);
            % Update the SSP for this time step based on SMV's orbital elements
            SSP_lat =  asin(sin_inc*sin(tt));
            SSP_long = mod(atan(cos_inc*tan(tt)) + raan_now,2*pi);
            if mod(tt,2*pi) > pi/2 && mod(tt,2*pi) < 3*pi/2;
                SSP_long = mod(SSP_long - pi,2*pi);
            end
            SSP(count,1) = SSP_long*(180/pi);
            SSP(count,2) = SSP_lat*(180/pi);
            count = count + 1;
        end
    hold on;
    figure(2); plot(SSP(:,1),SSP(:,2),'k.')
    str2 = ['SMVs= ' num2str(losats) ', Coverage = ' num2str(earth_coverage) ...
            ', L/D= ' num2str(LDVALUE) ', \phi= ' num2str(max_lat*180/pi) ...
            ', inc= ' num2str(inc_now*(180/pi)) '\circ'];
    axis([0 360 -90 90]); grid off; box on;
    xlabel('Longitude'); ylabel('Latitude'); text(180,-
85,str2,'HorizontalAlignment','center','BackgroundColor','w','EdgeColor','k');
end
```

Constellation Fitness Function

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% THIS FUNCTION DETERMINES THE FITNESS OF A CONSTELLATION BY
% COMBINING THE  NUMBER OF CAVS WITH EARTH COVERAGE.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [fitness,earth_coverage] = main_fitness_func_bin(constellation,gen)
global earth_grid lambda grid_points A fitfun
global m_motion TTT big_array slope yint h k
global maxtheta maxtime1 x1 y1 z1 maxgen bigrows

% Apply appropriate model for exponent
% Linear
if fitfun == 1;
    n = slope*gen + yint;
end
% Right parabola
if fitfun == 2;
```

```
    n = sqrt((gen - h)/A) + k;
end
% Up parabola
if fitfun == 3;
    n = A*(gen - h)^2 + k;
end

% Reshape the chromosome string into a matrix representing the
% constellation to be evaluated
vect1 = bin2dec(reshape(gs_unblank(constellation),20,100)')';

if sum(vect1,2) == 0;
    num_sats = 1;
    earth_coverage = 0.1;
    fitness = num_sats/(earth_coverage^n);
    return
end

vect1 = vect1';
chr1_const = big_array(vect1(find(vect1),:),:);
[num_sats,nothing] = size(chr1_const);
earth_grid(:,3) = 0;
for chrom_count = 1:num_sats;
    if chr1_const(chrom_count,4) == 1;
        % Get orbital elements for next SMV
        inc_now = chr1_const(chrom_count,1);
        raan_now = chr1_const(chrom_count,2);
        arglat_now = chr1_const(chrom_count,3);
        % Calculate values outside loop to improve speed
        cos_inc = cos(inc_now);
        sin_inc = sin(inc_now);
        swath_test = cos(lambda);
        % Set stepsize for propagation loop
        tt_incr = lambda;
        % Increment footprint until 90 minute limit
        for tt = arglat_now + maxtheta*(pi/180):tt_incr:arglat_now + m_motion*(TTT -
                maxtime1)+ maxtheta*(pi/180);
            % Update the SSP for this time step based on SMV's orbital elements
            SSP_lat = pi/2 - asin(sin_inc*sin(tt));
            SSP_long = mod(atan(cos_inc*tan(tt)) + raan_now,2*pi);
            if mod(tt,2*pi) > pi/2 && mod(tt,2*pi) < 3*pi/2;
                SSP_long = mod(SSP_long - pi,2*pi);
            end
            x2 = cos(SSP_long)*sin(SSP_lat);
            y2 = sin(SSP_long)*sin(SSP_lat);
```

```
            z2 = cos(SSP_lat);
            % Perform angular distance check of grid point and update grid coverage
            test1 = (x1*x2 + y1*y2 + z1*z2);
            indx1 = find(test1 >= swath_test);
            earth_grid(indx1,3) = earth_grid(indx1,3) + 1;
        end
    end
end
% Calculate Earth coverage
coverage_counter = 0;
xtra_cov = 0;
for cc = 1:grid_points;
    if earth_grid(cc,3) >= 1;
        coverage_counter = coverage_counter + 1;
    end
    xtra_cov = xtra_cov + earth_grid(cc,3);
end
earth_coverage = coverage_counter/grid_points;
extra_coverage = xtra_cov/grid_points;

% Return fitness for this constellation
if earth_coverage >= .5;
    fitness = num_sats/(earth_coverage^n);
else
    fitness = num_sats/(.1^n);
end
```

Bibliography

1. Spacy, William L. II, *Does the United States Need Space-Based Weapons?*, The Cadre Papers, September 1999.

2. Mason, William J., Coverstone-Carroll, Victoria, Hartmann, John W., *Optimal Earth Orbiting Satellite Constellations Via A Pareto Genetic Algorithm*, AIAA/AAS Astrodynamics Specialist Conference and Exhibit, Boston, MA, Aug. 10-12, 1998, Collection of Technical Papers (A98-37348 10-13), Reston, VA, American Institute of Aeronautics and Astronautics, 1998.

3. Luders, R. D., *Continuous Zonal Coverage: A Generalized Analysis*, Aerospace Corporation, 1974.

4. Luders, R. D., *Satellite Networks for Continuous Zonal Coverage*, American Rocket Society Journal, Vol. 31, February 1961.

5. Rider, L., *Analytic Design of Satellite Constellations for Zonal Earth Coverage Using Inclined Circular Orbits*, The Journal of the Astronautical Sciences, Vol. 34, No. 1, January-March 1966.

6. Walker, J.G., *Continuous Whole-Earth Coverage by Circular-Orbit Satellite Patterns*, Technical Report 77044, Royal Aircraft Establishment, March 24, 1977.

7. Adams, W.S., Hopkins, R. G., *Minimal Arbitrarily Phased Constellations With A Given Inclination Providing Single Global Coverage*, Advances in the Astronautical Sciences, American Astronautical Society, 1991.

8. Wertz, James R. and Larson, Wiley J., *Space Mission Analysis and Design, Third Edition*, Space Technology Library, 1999.

9. Chobotov, Vladimir A., *Orbital Mechanics Third Edition*, American Institute of Aeronautics and Astronautics, 2002.

10. *MATLAB*, Version 6.1, Release 12.1. Computer software, Copyright 1984-2001, The Math Works, Inc.

11. *Genetic Search Toolbox*, Version 1.0.4. Computer software, Copyright 1988-2003, Optimal Syntheis, Inc.

12. *Genetic Search Toolbox User's Manual*, Optimal Synthesis Inc., Palo Alto, CA, 1998-2003, pp 1-3.

13. Harpold, J.C., Graves, C.A., *Shuttle Entry Guidance*, AAS Paper 78-147, American Astronautical Society, Anniversary Conference, 25th, Houston, Tex., Oct. 30-Nov. 2, 1978.

14. Regan, Frank J., Anandakrishnan, Satya M., *Dynamics of Atmospheric Re-Entry*, American Institute of Aeronautics and Astronautics, 1993.

15. Hsu, Fu-Kuo and Kuo, Te-Son, *Complete Footprint of Lifting Reentry Vehicles*, Acta Astronautica , vol. 21, March 1990.

16. N. X. Vinh, A. Busemann, and R.D. Culp, *Hypersonic and Planetary Entry Flight Mechanics*, Univ. of Michigan Press, 1980, pp. 19-28, 345-356.

17. Holland, J.H., *Adaptation in Natural and Artificial Systems*, Cambridge, MA: The MIT Press, 1993.

18. Koza, J.R., *Genetic Programming*, Cambridge, MA: The MIT Press, 1992.

Vita

Major Jason Anderson was born in Des Moines, IA. After completing high school in Bemidji, MN, he attended Troy State University in Troy, AL, where he was awarded a B.S. Degree in Mathematics in 1992. He earned his Air Force commission through Officer Training School at Lackland AFB, TX in July 1993. His first assignment was to Falcon AFB, CO, where he was qualified as a Satellite Vehicle Operator for the MILSTAR satellite. Additionally, he trained as an Orbit Analyst for the MILSTAR, DSCSIII, and UHF F/O satellite programs. He then transferred to Malmstrom AFB, MT where he performed ICBM crew commander, ICBM crew evaluator, and wing senior evaluator duties in the MMIII ICBM weapon system. His next assignment was to 20AF Headquarters at F.E. Warren AFB, WY where he carried out staff officer, ICBM crew evaluator, and ICBM policy duties. His current assignment is at the Air Force Institute of Technology at Wright-Patterson AFB, OH where he is working towards a Masters Degree in Space Operations. He currently lives in Dayton, OH with his wife and children

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From – To)*
12 Mar 2004	**Master's Thesis**	Aug 2002 – Mar 2004

4. TITLE AND SUBTITLE

OPTIMAL CONSTELLATION DESIGN FOR ORBITAL MUNITIONS DELIVERY SYSTEM

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Anderson, Jason, Major, USAF

5d. PROJECT NUMBER
ENR# 2003-107

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GSS/ENY/04-M01

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Security Space Architect
Attn: Lt Col Edward Phillips
11242 Waples Mill Rd.
Chantilly, VA 20153 Comm 571-432-1300

10. SPONSOR/MONITOR'S ACRONYM(S)
NSSA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Rather than delivering conventional munitions through the airspace of uncooperative nations, a constellation of space-stored weapons could potentially target any point on the Earth and arrive within the time it takes to de-orbit and re-enter through the atmosphere. The research involves applying the dynamics of atmospheric re-entry to a Common Aero Vehicle (CAV) and defining a 'footprint' of attainable touchdown points. The footprint is moved forward to create a swath representing all the possible touchdown points in a 90 minute window. A nominal constellation of CAVs is established using a 'streets of coverage' technique, and both analytic studies and numeric genetic algorithm techniques are used to modify the nominal constellation. A minimum number of CAVs is identified which ensures payload delivery to an area of interest within 90 minutes.

15. SUBJECT TERMS
constellation design, atmospheric re-entry, genetic algorithm, common aero vehicle

16. SECURITY CLASSIFICATION OF: U			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
					Steven G. Tragesser
REPORT	**ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(Include area code)*
U	U	U	UU	85	(937) 255-6565, ext 4286; e-mail: steven.tragesser@afit.edu

Standard Form 298 (Rev: 8-98)
Prescribed by ANSI Std. Z39-18